



HAL
open science

Controller synthesis & Ordinal Automata

Thierry Cachat

► **To cite this version:**

| Thierry Cachat. Controller synthesis & Ordinal Automata. 2006. hal-00019897v1

HAL Id: hal-00019897

<https://hal.science/hal-00019897v1>

Preprint submitted on 3 Jun 2006 (v1), last revised 30 Oct 2006 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Controller synthesis & Ordinal Automata^{*}

Thierry Cachat

LIAFA/CNRS UMR 7089 & Université Paris 7, France

Abstract. Ordinal automata are used to model physical systems with Zeno behavior. Using automata and games techniques we solve a control problem formulated and left open by Demri and Nowak in 2005. It involves partial observability and a new synchronization between the controller and the environment.

1 Introduction

Controller synthesis. The synthesis of controller is today one of the most important challenges in computer science. Since [RW89] different formalisms have been considered to model (un)controllable and (un)observable actions. The problem is well understood for finite systems admitting infinite behavior (indexed by ω) [PR89]. Recent developments concern extensions to e.g. infinite state systems or timed systems [BDMP03].

Transforming control problems into two-player games have provided efficient solutions [Tho95]. In this setting the controller is modeled by a player and the environment by her opponent. Determining whether a controller exists falls down to determine the winner and computing a winning strategy is equivalent to synthesizing a controller.

Systems with Zeno behaviors. When modeling physical systems we face the problem that different components can have different time scales. For example the controller of an anti-lock braking system (ABS) is supposed to react much quicker than the physical environment. In the opposite one can consider physical systems admitting Zeno behavior —infinitely many actions in a finite amount of time— whereas the controller is a computer with constant clock frequency. A simple example is a bouncing ball. Another one is the physical description of an electronic circuit which evolves much quicker than its logical description in VHDL. The speeds are so different that one can consider that the former one evolves infinitely quicker than the latter one.

Following this idea Demri and Nowak [DN05] have proposed to model physical systems by ordinal automata, thus admitting ordinal sequences as behavior (typically of length ω^k). They define a logic $LTL(\omega^k)$ as an extension of LTL to express properties of such systems. The controller should be a usual automaton whose execution is an ω -sequence. The synchronization between controller

^{*} The author acknowledges partial support by the ACI “Sécurité et Informatique” CORTOS. <http://www.lsv.ens-cachan.fr/aci-cortos/>

and environment is the following: environment makes ω^{k-1} steps “alone”, then controller and environment makes one step together, and so on.

Particularly in the context of timed systems, different techniques have been proposed to forbid or restrict Zeno behaviors, see introduction of [AFH⁺03] for an overview. Our claim is that we want to allow Zeno behavior, to model them and express properties about them, and finally to control such systems.

Our contribution. The main contribution of our article is a solution to the control problem stated and left open in [DN05]. Given a physical system modeled by an ordinal automaton and a formula ψ of $\text{LTL}(\omega^k)$ we want to determine whether a controller exists and synthesize one. The technique used is to transform the control problem into a game problem. Because of the unobservable actions and also because of the different time scales, the controller can not fully observe the current state of the system. For that reason we construct a game of imperfect information. Another difficulty is that the length of the interaction is greater than ω , but fortunately one can summarize ω^{k-1} steps done by the environment “alone”. Several games and automata techniques are used.

Related work. It is known that games of imperfect information have higher computational complexity [Rei84]. Zeno behavior have already been considered in the literature. In [BP00] languages of ordinal words accepted by timed automata are studied. In the framework of hybrid systems [AM98,Bou99] or cellular automata on continuous time and space [DL05] it is known that allowing Zeno behaviors gives rise to highly undecidable problems. In [DN05] Demri and Nowak solve the satisfiability and the model-checking problem for $\text{LTL}(\omega^k)$: given an ordinal automaton reading ω^k -sequences and a formula ψ , determine whether every run of the automaton satisfies ψ . For this they use a “succinct” form of ordinal automata to have better complexity bounds.

Plan of the paper In the next section we present the temporal logic $\text{LTL}(\omega^k)$, ordinal automata and the control problem. We show a translation to first order logic. In section 3 we solve our main problem. We first explain how to translate it to a game and why the controller has imperfect information about the system.

2 Reasoning about transfinite sequences

We assume basic knowledge about ordinals less than ω^ω , see e.g. [Ros82]. An *ordinal* is a well and totally ordered set. It is either 0 or a successor ordinal of the form $\beta + 1$ or a limit ordinal. The first limit ordinal is denoted ω . For all ordinal α , $\beta < \alpha \Leftrightarrow \beta \in \alpha$ and $\alpha = \{\beta : \beta < \alpha\}$. In this article we restrict ourselves to ordinals less or equal than ω^ω . By the Cantor Normal Form theorem, for all $\alpha < \omega^\omega$ there exists unique integers p, n_1, \dots, n_p and k_1, \dots, k_p such that $k_1 > k_2 > \dots > k_p$ and $\alpha = \omega^{k_1} n_1 + \omega^{k_2} n_2 + \dots + \omega^{k_p} n_p$. Recall e.g. that $2\omega = \omega$ and $\omega + \omega^2 = \omega^2$. An ordinal α is said to be closed under addition whenever $\beta, \beta' < \alpha$ implies $\beta + \beta' < \alpha$. In particular for every $\alpha \leq \omega^\omega$, α is closed under

addition iff α is equal to ω^β for some $\beta \leq \omega$ or $\alpha = 0$. In the following we will consider a logic whose models are ω^k sequences for some $k < \omega$.

2.1 Temporal Logic

We recall the definition of the logic $\text{LTL}(\alpha)$ introduced in [DN05]. For every ordinal α closed under addition, the models of $\text{LTL}(\alpha)$ are precisely sequences of the form $\sigma : \alpha \rightarrow 2^{\text{AP}}$ for some countably infinite set AP of atomic propositions. The formulas of $\text{LTL}(\alpha)$ are defined as follows: $\phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \mathbf{X}^\beta\phi \mid \phi_1\mathbf{U}^{\beta'}\phi_2$, where $p \in \text{AP}$, $\beta < \alpha$ and $\beta' \leq \alpha$. The satisfaction relation is inductively defined below where σ is a model for $\text{LTL}(\alpha)$ and $\beta < \alpha$:

- $\sigma, \beta \models p$ iff $p \in \sigma(\beta)$,
- $\sigma, \beta \models \phi_1 \wedge \phi_2$ iff $\sigma, \beta \models \phi_1$ and $\sigma, \beta \models \phi_2$, $\sigma, \beta \models \neg\phi$ iff not $\sigma, \beta \models \phi$,
- $\sigma, \beta \models \mathbf{X}^{\beta'}\phi$ iff $\sigma, \beta + \beta' \models \phi$,
- $\sigma, \beta \models \phi_1\mathbf{U}^{\beta'}\phi_2$ iff there is $\gamma < \beta'$ such that $\sigma, \beta + \gamma \models \phi_2$ and for every $\gamma' < \gamma$, $\sigma, \beta + \gamma' \models \phi_1$.

Closure under addition of α guarantees that $\beta + \beta'$ and $\beta + \gamma$ above are strictly smaller than α . Usual LTL is expressively equivalent to $\text{LTL}(\omega)$: \mathbf{X} is equivalent to \mathbf{X}^1 and \mathbf{U} is equivalent to \mathbf{U}^ω , conversely \mathbf{X}^n and \mathbf{U}^n can be expressed in LTL. Standard abbreviations are also extended: $\mathbf{F}^\beta\phi \stackrel{\text{def}}{=} \top\mathbf{U}^\beta\phi$ and $\mathbf{G}^\beta\phi \stackrel{\text{def}}{=} \neg\mathbf{F}^\beta\neg\phi$. Using Cantor Normal Form it is easy to effectively encode an $\text{LTL}(\omega^k)$ formula for $k < \omega$. We provide below properties dealing with limit states that can be easily expressed in $\text{LTL}(\omega^k)$ ($k \geq 2$).

“ p holds in the states indexed by limit ordinals strictly less than ω^k ”:

$$\mathbf{G}^{\omega^k}(\mathbf{X}^\omega p \wedge \dots \wedge \mathbf{X}^{\omega^{k-1}} p).$$

For $1 \leq k' \leq k - 2$, “if p holds infinitely often in states indexed by ordinals of the form $\omega^{k'} \times n$, $n \geq 1$, then q holds in the state indexed by $\omega^{k'+1}$ ”:

$$(\mathbf{G}^{\omega^{k'+1}} \mathbf{F}^{\omega^{k'+1}} \mathbf{X}^{\omega^{k'}} p) \Rightarrow (\mathbf{X}^{\omega^{k'+1}} q).$$

2.2 Translation to First Order Logic

In [DN05] it is proved that $\text{LTL}(\omega^\omega)$ (hence also $\text{LTL}(\omega^k)$) can be translated to the monadic second order theory of $\langle \omega^\omega, < \rangle$, which gives a non-elementary decision procedure for satisfiability [BS73]. We improve this result by showing that $\text{LTL}(\omega^\omega)$ can be translated even to the first order theory (FO) of $\langle \omega^\omega, < \rangle$.

Proposition 1. *For every $\text{LTL}(\omega^\omega)$ formula there exists an equivalent first order formula over $\langle \omega^\omega, < \rangle$.*

It is open whether the converse also holds, extending Kamp’s theorem [Kam68].

Proof (sketch). The main point is the definition of a formula $+_{\beta}(x, y)$ for some $\beta < \omega^{\omega}$ such that $\langle \omega^{\omega}, < \rangle \models_v +_{\beta}(x, y)$ with $v : \{x, y\} \rightarrow \omega^{\omega}$ iff $v(y) = v(x) + \beta$. The relation \models_v is the standard satisfaction relation under the valuation v . The formulas of the form $+_{\beta}(x, y)$ with $\beta < \omega^{\omega}$ are inductively defined as:

1. $+_0(x, y) \stackrel{\text{def}}{=} (x = y)$,
2. $+_1(x, y) \stackrel{\text{def}}{=} \forall z (z > x \Rightarrow y \leq z) \wedge (x < y)$,
3. $+_{\omega^k n + \beta}(x, y) \stackrel{\text{def}}{=} \exists z +_{\omega^k}(x, z) \wedge +_{\omega^k(n-1) + \beta}(z, y)$ ($n \geq 1, k \geq 0$),
4. $+_{\omega^k}(x, y) \stackrel{\text{def}}{=} (x < y) \wedge \forall z (x < z \leq y \Rightarrow \exists z' (+_{\omega^{k-1}}(z, z') \wedge z' < y)) \wedge \forall y' [(x < y') \wedge \forall z (x < z \leq y' \Rightarrow \exists z' (+_{\omega^{k-1}}(z, z') \wedge z' < y')) \Rightarrow y \leq y']$ ($k \geq 1$).

For $k = 1$, the latter formula is written in the following way. The ordinal y such that $+_{\omega}(x, y)$ holds is greater than x , greater than every finite step successors of x , and y is the least ordinal satisfying this two conditions. By induction one can show that $y > x + n$ for every $n < \omega$. Analogously for $k > 1$, the formula implies that $y > x + \omega^{k-1}n$ for every $n < \omega$. \square

The monadic second order theory of $\langle \omega^{\omega}, < \rangle$ has a non-elementary decision procedure. For that reason we use ordinal automata, both to model physical systems and to represent specifications.

2.3 Ordinal Automata

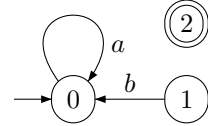
A particular class of ordinal automata is well suited to solve our problem. See [Bed98] for the equivalence between different definitions. An ordinal automaton is a tuple $(Q, \Sigma, \delta, E, I, F)$ where:

- Q is a finite set of states,
- Σ is a finite alphabet,
- $\delta \subseteq Q \times \Sigma \times Q$ is a one-step transition relation,
- $E \subseteq 2^Q \times Q$ is a limit transition relation,
- $I \subseteq Q$ is a finite set of initial states,
- $F \subseteq Q$ is a finite set of final states.

We write $q \xrightarrow{a} q'$ whenever $\langle q, a, q' \rangle \in \delta$ and $P \rightarrow q$ whenever $\langle P, q \rangle \in E$. A *path* of length $\alpha + 1$ is an $(\alpha + 1)$ -sequence $r : \alpha + 1 \rightarrow Q$ labeled by an α -sequence $\sigma : \alpha \rightarrow \Sigma$ such that for every $\beta \in \alpha$, $r(\beta) \xrightarrow{\sigma(\beta)} r(\beta + 1)$ and for every limit ordinal $\beta \in \alpha + 1$, there is $P \rightarrow r(\beta) \in E$ s.t. $P = \text{cofinal}(\beta, r)$ with $\text{cofinal}(\beta, r) \stackrel{\text{def}}{=} \{q \in Q : \text{for every } \gamma \in \beta, \text{ there is } \gamma' \text{ such that } \gamma < \gamma' < \beta \text{ and } r(\gamma') = q\}$.

If moreover $r(0) \in I$, it is a *run*. If moreover $r(\alpha) \in F$, it is accepting.

Example 1. We present here an example of ordinal automaton \mathcal{A} with limit transitions $\{0\} \rightarrow 1$ and $\{0, 1\} \rightarrow 2$. One can show that $L(\mathcal{A})$ contains only ω^2 -sequences and $L(\mathcal{A}) = (a^{\omega} \cdot b)^{\omega}$.



For all $k < \omega$ there exists an ordinal automaton accepting exactly the sequences of length ω^k , using $k + 1$ states. But if an ordinal automaton accepts a sequence of length ω^ω , then it must also accept longer sequences. That is a second reason, beside closure under addition, why we restrict ourselves to ordinals less than ω^ω .

Level An ordinal automaton $\mathcal{A} = \langle Q, \Sigma, \delta, E, I, F \rangle$ is of *level* $k \geq 1$ iff there is a map $l : Q \rightarrow \{0, \dots, k\}$ such that:

- for every $q \in F$, $l(q) = k$;
- $q \xrightarrow{a} q' \in \delta$ implies $l(q') = 0$ and $l(q) < k$;
- $P \rightarrow q \in E$ implies $l(q) \geq 1$, for every $q' \in P$, $l(q') < l(q)$, and there is $q' \in P$ such that $l(q') = l(q) - 1$.

The idea is that a state of level i is reached at positions $\beta + \omega^i \cdot j$, $j < \omega$. Since [VW86], different techniques for translating logic formulas to automata are widely used.

Proposition 2 ([DN05]). *For all LTL(ω^k) formula, there exists an equivalent ordinal automaton.*

This result can be obtain by translating an LTL(ω^k) formula into a equivalent first order formula (or even monadic second order) and applying results from [BS73]. In [DN05] a succinct version of ordinal automata is defined to improve the complexity of the translation from non-elementary to polynomial (reps. exponential) when integers in the formulas are encoded in unary (resp. binary).

2.4 Control Problem

Before we recall the control problem from [DN05] we need some preliminary definitions. In order for the physical system to evolve much faster than the controller we need a particular synchronization between them.

Synchronous product. We define below the synchronous product of two ordinal automata having possibly different alphabets. They synchronize only on the common actions. This is used later to model unobservable actions. Let $\Sigma_i = 2^{Act_i}$ for $i = 1, 2$, a letter from Σ_i is a set of actions. Given two ordinal automata $\mathcal{A}_i = \langle Q_i, \Sigma_i, \delta_i, E_i, I_i, F_i \rangle$, for $i = 1, 2$, their synchronous product is defined as $\mathcal{A}_1 \times \mathcal{A}_2 = \langle Q, \Sigma, \delta, E, I, F \rangle$ where:

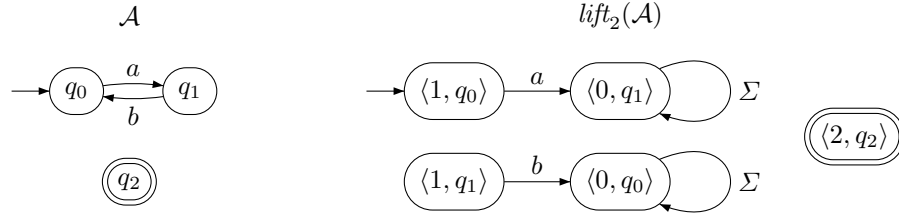
- $Q = Q_1 \times Q_2$, $\Sigma = 2^{Act_1 \cup Act_2}$.
- $\langle q_1, q_2 \rangle \xrightarrow{a} \langle q'_1, q'_2 \rangle \in \delta$ iff $q_1 \xrightarrow{a \cap Act_1} q'_1$ and $q_2 \xrightarrow{a \cap Act_2} q'_2$.
- $P \rightarrow \langle q_1, q_2 \rangle \in E$ iff there exists $P_1 \rightarrow q_1 \in E_1$ and $P_2 \rightarrow q_2 \in E_2$ such that $\{q : \langle q, q' \rangle \in P\} = P_1$ and $\{q' : \langle q, q' \rangle \in P\} = P_2$.
- $I = I_1 \times I_2$, $F = F_1 \times F_2$.

Lifting. In order to synchronize the system with a controller working on ω -sequences, we need to transform the controller so that its product with \mathcal{S} only constraints states on positions $\omega^{k-1} \times n$, $n < \omega$. The other positions are not constrained.

Let $\mathcal{A} = \langle Q, \Sigma, \delta, E, I, F, l \rangle$ be an automaton of level 1. We define its lifting $lift_k(\mathcal{A})$ at level $k \geq 2$ to be the automaton $\langle Q', \Sigma, \delta', E', I', F', l' \rangle$ by:

- $Q' = \{0, \dots, k\} \times Q$, $I' = \{k-1\} \times I$, $F' = \{k\} \times F$
- $l'(\langle i, q' \rangle) = i$,
- $\delta' = \{ \langle k-1, q \rangle \xrightarrow{a} \langle 0, q' \rangle : q \xrightarrow{a} q' \in \delta \} \cup \{ \langle i, q \rangle \xrightarrow{a} \langle 0, q \rangle : 0 \leq i \leq k-2, a \in \Sigma, q \notin F \}$,
- $E' = \{ \{ \langle 0, q \rangle, \dots, \langle i-1, q \rangle \} \rightarrow \langle i, q \rangle : 1 \leq i < k, q \in Q \} \cup \{ \{ \langle 0, q_1 \rangle, \dots, \langle k-1, q_1 \rangle, \dots, \langle 0, q_n \rangle, \dots, \langle k-1, q_n \rangle \} \rightarrow \langle k, q \rangle \mid \{q_1, \dots, q_n\} \rightarrow q \in E \}$.

Example 2. We present below an example of ordinal automaton \mathcal{A} with limit transition $\{q_0, q_1\} \rightarrow q_2$ and the corresponding automaton $lift_2(\mathcal{A})$ with limit transitions $\{\langle 0, q_0 \rangle\} \rightarrow \langle 1, q_0 \rangle$, $\{\langle 0, q_1 \rangle\} \rightarrow \langle 1, q_1 \rangle$, and $\{\langle 0, q_0 \rangle, \langle 1, q_0 \rangle, \langle 0, q_1 \rangle, \langle 1, q_1 \rangle, \} \rightarrow \langle 2, q_2 \rangle$. We omit useless transitions.



Proposition 3 ([DN05]). For all $w \in \Sigma^{\omega^k}$, $w \in L(lift_k(\mathcal{A}))$ iff the word $w' \in \Sigma^{\omega}$, defined by $w'(i) = w(\omega^{k-1} \times i)$, is in $L(\mathcal{A})$.

A physical system \mathcal{S} is modeled as a structure

$$\langle \mathcal{A}_{\mathcal{S}}, Act_c, Act_o, Act \rangle$$

where $\mathcal{A}_{\mathcal{S}}$ is an ordinal automaton of level k with alphabet 2^{Act} where Act is a finite set of actions, $Act_o \subseteq Act$ is the set of observable actions, $Act_c \subseteq Act_o$ is the set of controllable actions. The set $Act \setminus Act_c$ of uncontrollable actions is denoted by Act_{nc} . A specification of the system \mathcal{S} is naturally an LTL(ω^k) formula ψ . A controller \mathcal{C} for the pair $\langle \mathcal{S}, \psi \rangle$ is a system whose complete executions are ω -sequences (typically ordinal automata of level 1) verifying the properties below.

- Only observable actions are present in the controller. Hence, thanks to the synchronization mode, in the product system between \mathcal{S} and \mathcal{C} , unobservable actions do not change the \mathcal{C} -component of the current state. So the alphabet of \mathcal{C} is 2^{Act_o} .
- From any state of \mathcal{C} , uncontrollable actions can always be executed: $\forall q \cdot \forall a \subseteq Act_o \setminus Act_c$, there is a transition $q \xrightarrow{b} q'$ in \mathcal{C} such that $b \cap Act_{nc} = a$.

- Finally, the system \mathcal{S} controlled by \mathcal{C} satisfies ψ . Because \mathcal{S} and \mathcal{C} work on sequences of different length, the controlled system is in fact equal to $\text{lift}_k(\mathcal{C}) \times \mathcal{S}$. So $\text{lift}_k(\mathcal{C}) \times \mathcal{S} \models \psi$ should hold. This is equivalent to the emptiness of the language of the product automaton $\text{lift}_k(\mathcal{C}) \times \mathcal{S} \times \mathcal{A}_{\neg\psi}$.

We say that \mathcal{C} is a controller for \mathcal{S} (without mentioning ψ) if \mathcal{C} fulfills the first two conditions. The notion of final state is not relevant for the controller or the physical system. To conform with previous definitions we require that every $(\omega + 1)$ -run of the controller and $(\omega^k + 1)$ -run of \mathcal{S} end in a final state.

3 Solving the Control Problem

Given a physical system \mathcal{S} modeled by an ordinal automaton $\mathcal{A}_{\mathcal{S}}$ of level k and an LTL(ω^k)-formula ψ , we are looking for a controller \mathcal{C} such that $\text{lift}_k(\mathcal{C}) \times \mathcal{A}_{\mathcal{S}} \models \psi$ and \mathcal{C} has the expected properties about uncontrollable and unobservable actions.

From Control Problem to Game. Let $\mathcal{B} = \text{lift}_k(\mathcal{C}) \times \mathcal{A}_{\mathcal{S}} \times \mathcal{A}_{\neg\psi}$. At a given point in a run of \mathcal{B} the controller is in a state q . From q and for all $o \subseteq \text{Act}_o \cap \text{Act}_{nc}$ it must have at least one transition labeled by $o \cup c$ for some $c \subseteq \text{Act}_c$. The most general form of a controller (possibly with infinite memory) is a function $f : (2^{\text{Act}_o})^* \times (2^{\text{Act}_o \cap \text{Act}_{nc}}) \rightarrow 2^{\text{Act}_c}$, because the current state of the controller shall only depend on the past observable actions. This function is exactly a strategy in a game that we will define. A controller for $\langle \mathcal{S}, \psi \rangle$ is such that every run according to f is winning.

Let $\mathcal{A} = \mathcal{A}_{\mathcal{S}} \times \mathcal{A}_{\neg\psi}$. It is also an ordinal automaton of level k : $\mathcal{A} = \langle Q, \Sigma, \delta, E, I, F, l \rangle$. We are looking for a controller \mathcal{C} such that $\text{lift}_k(\mathcal{C}) \times \mathcal{A}$ is empty. We will consider a game where the environment tries to built an accepting run of \mathcal{A} , whereas the controller tries to avoid that, using the controlled actions. In fact the environment plays both for the system \mathcal{S} and for the automaton of $\neg\psi$, as we will see later.

3.1 Some Definitions from Game Theory

We recall some definitions about games. See for example [Tho95,GTW02] for an introduction. An *arena*, or *game graph*, is a triple (V_0, V_1, G) , where $V = V_0 \cup V_1$ is the set of vertices and $G \subseteq V \times V$ is the set of edges. The vertices of V_0 belongs to Player 0, those of V_1 to Player 1 ($V_0 \cap V_1 = \emptyset$). A *play* from $v_0 \in V$ proceeds as follows: if $v \in V_0$, Player 0 chooses a successor v_1 of v_0 , else Player 1 does. Again from $v_1 \in V_i$, Player i chooses a successor v_2 of v_1 , and so on.

A play $\pi = v_0, v_1, v_2, \dots$ is a finite or infinite sequence of vertices such that $\forall i, (v_i, v_{i+1}) \in G$. If the play is finite, the convention is that the player who belongs the last vertex loses (he is stuck). If the play is infinite, the winner is determined by a *winning set*, $Win \subseteq V^\omega$: Player 0 wins an infinite play π if and only if $\pi \in Win$. Usually Win is an ω -regular set, defined by a Büchi,

Rabin, parity or Muller automaton. One speaks also of *winning condition*. A *game* (V_0, V_1, G, Win) is an arena together with a winning condition and possibly an initial vertex $v_0 \in V$.

For a game or an automaton, a Büchi condition is given by a set $F \subseteq V$ of “final” vertices and $\pi \in Win$ if and only if $\forall i > 0, \exists j > i, \pi_i \in F$. A Muller condition is given by $\mathcal{F} \subseteq 2^V$, $\mathcal{F} = \{F_1, \dots, F_n\}$, and $\pi \in Win$ if and only if the set of states visited infinitely often along π is equal to one of the F_i ’s.

A *strategy* for Player 0 is a (partial) function $f_0 : V^*V_0 \mapsto V$ such that for every prefix $v_0, v_1, v_2, \dots, v_i$ of a play, where $v_i \in V_0$, $f(v_0v_1v_2 \dots v_i)$ is a vertex v_{i+1} such that $(v_i, v_{i+1}) \in G$. A play π is played according to a strategy f_0 if $\forall i, v_i \in V_0 \Rightarrow v_{i+1} = f(v_0v_1v_2 \dots v_i)$. A strategy for Player 1 is defined analogously. A strategy of Player 0 is *winning* if every play according to it is winning for Player 0. An important case in practice is when the strategy is *positional*: it depends only on the current vertex, not on the past of the play, *i.e.*, for all $v_0, v_1, v_2, \dots, v_i$, $f(v_0v_1v_2 \dots v_i) = f(v_i)$.

From [Mar75] we know that every zero-sum two-player turn based game of complete information with Borel winning condition (including ω -regular and many more) is determined: from a given initial configuration, one of the players has a winning strategy.

In the case of incomplete information, the players do not in general know exactly the current position of the game. They only know that the position belongs to a certain set of uncertainty. The move chosen by a player (by his strategy) shall depend on this set, but not on the precise position of the play. As we will see in some cases one can transform such a game into a game of complete information, where a vertex represents a set of positions of the original game.

3.2 Optimistic Version: the Control has Complete Information

In this section we introduce a game (\mathcal{G}_1) modeling the interaction between the controller and the environment. This game does not yet allow to give a solution to our control problem, but it can help understand the more complicated game (\mathcal{G}_2) defined in the next section.

From the definition of *lift_k* we see that the controller can act only every ω^{k-1} steps of the environment. Our aim is to summarize ω^{k-1} steps of the environment in a single step. One can compute a relation $\mathcal{R} \subseteq Q \times 2^Q \times Q$ such that $(q, P, q') \in \mathcal{R}$ iff there exists in \mathcal{A} a path from q to q' of length $\omega^{k-1} + 1$ where the set of states seen along this path is exactly P . Note that to determine \mathcal{R} , one has to look for cycles in \mathcal{A} and states that are seen infinitely often, but in \mathcal{R} itself we only need to know states that are ever visited. Relation \mathcal{R} can be computed in time $2^{\mathcal{O}(|Q|)}$ [Car02].

Let \mathcal{G}_1 be the game between the environment (Env) and the controller (Cont) defined by the following steps:

1. Env chooses $q_0 \in I$, an initial state of \mathcal{A} ,
2. Env chooses $u \subseteq Act_{nc}$,
3. Cont chooses $c \subseteq Act_c$ (controllable actions),

4. Env chooses a one step transition $q_0 \xrightarrow{u \cup c} q'_0$ in \mathcal{A} ,
5. Env chooses P_0 and q_1 such that $(q'_0, P_0, q_1) \in \mathcal{R}$, summarizing ω^{k-1} steps,
6. continue from q_1 : go to point 2 whit q_1 instead of q_0 .

A play of \mathcal{G}_1 is a finite or infinite sequence $q_0, q'_0, P_0, q_1, q'_1, P_1, \dots$ summarizing a run in \mathcal{A} . It can be finite only if Env is stuck in a deadlock, in this case Env loses. If the play is infinite, the winner is determined by \mathcal{A} : one has to determine whether the run can be followed by a transition to a final state (of level k) of \mathcal{A} . For that we have to know which states are present in infinitely many P_i 's.

Related to point 4, it is possible that \mathcal{A} is not deterministic. In particular it is possible that $\mathcal{A}_{\neg\psi}$ is not deterministic and Env has to choose which subformulas of $\neg\psi$ he wants to make true.

3.3 A Solution With Incomplete Information

The game \mathcal{G}_1 above is much easier for the controller than our original control problem. It is not possible in general for the controller to know exactly the current state of the game for several reasons.

- As Env chooses $u \subseteq Act_{nc}$, Cont can only observe the actions that are in Act_o (recall also that \mathcal{A} is not necessarily deterministic).
- Moreover Cont cannot know exactly the (q_i, P_i) chosen by Env, because they correspond to the ω^{k-1} steps done by the environment without control.
- Also Cont cannot know exactly the initial state chosen by Env.

In this section we will define a game \mathcal{G}_2 where Cont has partial information. A position of the game is a subset Q_i of Q , such that Cont knows that the current state of the system is in Q_i , but does not know which state exactly. The game is described by the following steps:

1. the initial position is $Q_0 = I$, the set of initial states of \mathcal{A} ,
2. Env chooses $o \subseteq Act_o \cap Act_{nc}$,
3. Cont chooses $c \subseteq Act_c$,
4. there is a one step transition to

$$Q'_0 = \{q' \in Q : \exists i \subseteq Act \setminus Act_o, \exists q \in Q_0, q \xrightarrow{c \cup o \cup i} q'\},$$

5. there is a jump to Q_1 , summarizing ω^{k-1} steps

$$Q_1 = \{q \in Q : \exists q' \in Q'_0, \exists (q', P, q) \in \mathcal{R}\},$$

6. continue from Q_1 at point 2.

In this game the knowledge of Cont about the current state is exactly what a controller can compute in the original problem, based on the observable actions. A play is essentially a sequence $Q_0, Q'_0, Q_1, Q'_1, \dots$ (a more precise definition of the game graph is given below) and now it is more intricate to determine the winner. The sequence $Q_0, Q'_0, Q_1, Q'_1, \dots$ represents the point of view of the

controller, and we call it an abstract play. Afterward Env has to choose inside this abstract play a concrete path $q_0, q'_0, q_1, q'_1, \dots$ such that $q_i \in Q_i, q'_i \in Q'_i$ and compatible to the sequence of choices of c_i 's and o_i 's. That is to say Env has to choose a sequence of elements $u_i \in Act \setminus Act_o$ and $(q'_i, P_i, q_i) \in \mathcal{R}$. Then we are back to a play (or path) similar to those of \mathcal{G}_1 , and we can determine the winner. Therefore the set of winning plays of Env can be defined by a *non deterministic* Muller automaton searching a concrete path, as we will see below, after we make some comments.

The advantage that Env plays “abstractly” the game, and selects a concrete path only afterward is not unfair. Again we want a controller that is secure, and we worry if the environment *could have* “won”. And in the case that the controller does not have a winning strategy, it does not necessarily mean that the environment has one, but it means that there is a risk that the environment wins. This is related to the fact that games of incomplete information are not determined in general: it is possible that no player has a winning strategy.

We now describe the automaton defining the set of winning plays and then the arena in more details. Note that the sequence $Q_0, Q'_0, Q_1, Q'_1, \dots$ above is uniquely determined by the sequence $o_0, c_0, o_1, c_1, \dots$ of actions chosen by Cont and Env. The state space of the automaton \mathcal{A}_{Win} recognizing the winning plays for Env is $Q \times 2^Q$. For all $P \neq \emptyset$ there is a transition $(q, P) \xrightarrow{c \cup o} (q', \emptyset)$ if and only if $\exists i \subseteq Act \setminus Act_o, \exists q \xrightarrow{c \cup o \cup i} q'$ in \mathcal{A} and there is a transition $(q', \emptyset) \xrightarrow{c} (q, P)$ if and only if $\exists (q', P, q) \in \mathcal{R}$.

The automaton \mathcal{A}_{Win} non-deterministically guesses a run in \mathcal{A} conforming to the sequence $o_0, c_0, o_1, c_1, \dots$. The acceptance condition of \mathcal{A}_{Win} is then the same as those of \mathcal{A} : it can be seen as a Muller condition depending on the states appearing infinitely often in a run. It is given by a set of sets of states \mathcal{F} . The usual way to handle such a non-deterministic Muller automaton is to transform it into a non-deterministic Büchi automaton [GTW02, Ch. 1]. The Büchi automaton \mathcal{B}_{Win} simulates \mathcal{A}_{Win} and guesses at some point which subset of states are going to be visited infinitely often and that other states are no longer visited. The state space of \mathcal{B}_{Win} is $Q \cup Q \times \mathcal{F} \times (Q \cup \{q_f\})$. It checks in turn that each state of the chosen acceptance component $F \in \mathcal{F}$ is visited infinitely often and it is not necessary to remember the whole $(q, P) \in Q \times 2^Q$ of \mathcal{A}_{Win} .

Using e.g. Safra’s construction [GTW02, Ch. 3] one can transform the Büchi automaton \mathcal{B}_{Win} into a *deterministic* Rabin automaton \mathcal{C}_{Win} . Then the Index Appearance Record allows to have a deterministic parity automaton \mathcal{D}_{Win} [GTW02, p.86] [Löd98].

For defining the arena, we see that Cont and Env essentially choose the actions c_i and o_i :

$$\begin{aligned} V_{Env} &= 2^{Act_c} & V_{Cont} &= 2^{Act_o \cap Act_{nc}} \\ G &= (V_{Env} \times V_{Cont}) \cup (V_{Cont} \times V_{Env}) \end{aligned}$$

Now the product of the arena (V_{Env}, V_{Cont}, G) by the parity automaton \mathcal{D}_{Win} gives rise to a parity game on a finite graph. In fact the successive sets $Q_0, Q'_0, Q_1, Q'_1, \dots$

of the above description are computed by \mathcal{D}_{Win} (thanks to Safra's construction already in \mathcal{C}_{Win}). The sizes, in number of states, are as follows:

$$|\mathcal{C}_{Win}| = \mathcal{O}\left(2^{|\mathcal{B}_{Win}|}\right) = \mathcal{O}\left(2^{|\mathcal{Q}|^2 \cdot |\mathcal{F}|}\right) \quad \text{where } |\mathcal{F}| < 2^{|\mathcal{Q}|}$$

$$|\mathcal{D}_{Win}| = |\mathcal{C}_{Win}| 2^{\mathcal{O}(|\mathcal{B}_{Win}|)} \quad \text{hence} \quad |\mathcal{D}_{Win}| = 2^{\mathcal{O}(|\mathcal{B}_{Win}|)}$$

The size of \mathcal{D}_{Win} is exponential only in the number of Rabin pairs of the acceptance condition of \mathcal{C}_{Win} , which is in $\mathcal{O}(|\mathcal{B}_{Win}|)$. The best known deterministic algorithm for solving parity games is polynomial in the size of the graph, and exponential in the number of priorities. It computes a positional winning strategy for the winner. The time complexity is in

$$\mathcal{O}\left(\left(2^{|\mathcal{Act}_o|} \cdot |\mathcal{D}_{Win}|\right)^{|\mathcal{D}_{Win}|}\right)$$

Theorem 1. *The control problem defined in Section 2.4 can be solved in 3EXPTIME. Moreover if a controller exists, then there is one with finite memory of size in 2EXPTIME.*

The number of states of the controller is at most $2^{|\mathcal{Act}_o|} 2^{2^{|\mathcal{Q}|}}$. The complexity is measured in the number $|\mathcal{Q}|$ of states of $\mathcal{A} = \mathcal{A}_{\mathcal{S}} \times \mathcal{A}_{-\psi}$. Recall that the usual control problem is 2EXPTIME-complete [PR89] in the size of the system and the length of the formula.

Proof (Sketch). It is sufficient to prove the following facts. If the game \mathcal{G}_2 is won by Cont then a controller for $\langle \mathcal{S}, \psi \rangle$ exists, and it can be constructed. Conversely if a controller for $\langle \mathcal{S}, \psi \rangle$ exists then \mathcal{G}_2 is won by Cont.

By construction a strategy for Cont in \mathcal{G}_2 is a finite state automaton with expected properties about (un)observable and (un)controllable actions. Moreover if that strategy is winning, it defines a controller for $\langle \mathcal{S}, \psi \rangle$: every run of $lift_k(\mathcal{C}) \times \mathcal{S}$ fulfills ψ .

Conversely, if a controller for $\langle \mathcal{S}, \psi \rangle$ exists, possibly with infinite memory, then this controller provides a winning strategy for Cont in \mathcal{G}_2 . From the analysis above we know that if there is a winning strategy, then there is a winning strategy with finite memory, and one can compute it. \square

4 Perspectives

It is open whether the upper bounds of Theorem 1 are tight, and whether one can find LTL-fragments or restrictions on the physical system such that the complexity of the control problem is lower.

Given a timed automaton, it is possible to determine whether it has Zeno behaviors. Our motivation is to extend the semantics such that after ω transitions there is a limit transition to a new control state and the new clock values are the limit of the former one's.

A Zeno behavior is not necessarily an ordinal sequence, it can be a more general linear ordering (see [BC05]). One should extend the results to this more general class of automata.

Acknowledgments. Great thanks to Stéphane Demri and David Nowak for many interesting discussions, helpful comments on previous versions and for their help.

References

- [AFH⁺03] L. de Alfaro, M. Faëlla, T. A. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. *CONCUR'03*, LNCS 2761, pp. 142–156. 2003.
- [AM98] E. Asarin and O. Maler. Achilles and the tortoise climbing up the arithmetical hierarchy. *JCSS* 57(3), pp. 389–398. 1998.
- [BC05] A. Bès and O. Carton. A kleene theorem for languages of words indexed by linear orderings. *DLT'05*, LNCS 3572, pp. 158–167. 2005.
- [BDMP03] P. Bouyer, D. D'Souza, P. Madhusudan, and A. Petit. Timed control with partial observability. *CAV'03*, LNCS 2725, pp. 180–192. 2003.
- [Bed98] N. Bedon. *Langages reconnaissables de mots indexés par des ordinaux*. PhD thesis, Université de Marne-la-Vallée. 1998.
- [Bou99] O. Bournez. Achilles and the tortoise climbing up the hyper-arithmetical hierarchy. *TCS*, 210(1):21–71. 1999.
- [BP00] B. Bérard and C. Picaronny. Accepting Zeno words: A way toward timed refinements. *Acta Informatica*, 37(1):45–81. 2000.
- [BS73] J. R. Buchi and D. Siefkes. *The monadic second order theory of all countable ordinals, Lect. Notes in Math.* 328 Springer. 1973.
- [Car02] O. Carton. Accessibility in automata on scattered linear orderings. *MFCS'02*, LNCS 2420, pp. 155–164. 2002.
- [DL05] J. Durand-Lose. Abstract geometrical computation for black hole computation (extended abstract). In *Machines, computations, and universality*, LNCS 3354, pp. 176–187. 2005.
- [DN05] S. Demri and D. Nowak. Reasoning about transfinite sequences (extended abstract). *ATVA'05*, LNCS 3707. 2005.
- [GTW02] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, LNCS 2500. 2002.
- [GW94] P. Godefroid and P. Wolper. A partial approach to model checking. *Inform. and Comput.*, 110(2):305–326. 1994.
- [Kam68] H. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California at Los Angeles, 1968.
- [Lö98] C. Löding. Methods for the transformation of omega-automata: Complexity and connection to second order logic. Master's thesis, Christian-Albrechts-University of Kiel, 1998.
- [Mar75] D. A. Martin. Borel Determinacy. *Annals of Math.*, 102:363–371, 1975.
- [PR89] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL'89*, pp. 179–190. ACM, 1989.
- [Rei84] J. H. Reif. The complexity of two-player games of incomplete information. *J. Comput. System Sci.*, 29(2):274–301. 1984.
- [Ros82] J. G. Rosenstein. *Linear orderings*. Academic Press Inc. 1982.
- [RW89] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of IEEE* 77(1), pp. 81–98. 1989.
- [Tho95] W. Thomas. On the synthesis of strategies in infinite games. *STACS'95*, LNCS 900, pp. 1–13. 1995.
- [VW86] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. *LICS'86*, pp. 332–344. 1986.