A Turn-Based Approach for Qualitative Time Concurrent Games

Serge Haddad, Didier Lime, Olivier H Roux

A Turn-Based Approach for Qualitative Time Concurrent Games

Serge Haddad[1], Didier Lime[2][0000−0001−9429−7586], Olivier H. Roux[2]

[1] ENS Paris-Saclay, LMF, CNRS, INRIA, Université Paris-Saclay, France
[2] École Centrale de Nantes, LS2N, CNRS, UMR 6004, France

Abstract. We address concurrent games with a qualitative notion of time with parity objectives. This setting allows to express how potential controllers interact with their environment and more specifically includes relevant features: transient states where the environment will eventually act, controller avoiding of an environment action either by an immediate controller action or by masking it, etc. In order to solve the controller synthesis in this framework, we design a linear-time building of a timeless turn-based game and show a close connection between strategies of the controller in the two games. Thus we reduce the synthesis problem to a standard problem of turn-based game with parity objectives establishing as a side effect that pure memoryless strategies are enough for winning. Moreover we introduce permissiveness for safety and reachability games as a criterion to choose between winning strategies and prove that one can compute a most permissive strategy (when it exists) in linear time.

1 Introduction

Games and controller synthesis. Finite games on graphs [13] are widely recognized as an adequate formalism to address problems such as controller synthesis on discrete event systems, originally expressed and studied within the theory of supervision [14,12,9]. The control problem can indeed be expressed as a game between two players representing respectively the controller and the environment. A controller for the system can be synthesized as a winning strategy for the controller player, when it exists.

Real-time controller synthesis. For real-time systems, a strict turned-based game is an unnatural model, since the controller and the environment may play concurrently leading to concurrent games [5,8,7]. Adding quantitative delays before playing actions is a way to select which action should be played. This results in formalisms called (concurrent) timed games [11,6], for which tools like UPPAAL-Tiga are available [4]. Nevertheless, the algorithmics of timed games is costly, and for instance, the mere existence of a controller is an EXPTIME-complete problem [10] and the resulting strategies can be very large [1]. Furthermore from a modelling point of view, often the exact timing constraints are unknown and indeed not needed to ensure the existence of a winning strategy.

Qualitative time concurrent games. To overcome these issues the authors of [2,3] have introduced a model of qualitative time concurrent game with the

following features: actions of the environment may occur immediately or require some non null unknown delay and transient states where when the controller chooses not to (or cannot) play, an action of the environment is guaranteed to eventually occur. Then they have designed polynomial time (ad-hoc) algorithms synthesising (when it exists) a controller for reachability and safety goals.

Our contribution. Our contribution is threefold. First we extend the model of [2] by allowing some actions of the environment to be blocked by the controller and considering parity objectives. Then we design a linear-time building of a timeless turn-based game and show a close (but not one-to-one) connection between strategies of the controller in the two games. Thus we reduce the synthesis problem to a standard problem of turn-based game with parity objectives establishing as a side effect that pure memoryless strategies are enough for winning. Finally we introduce permissiveness for safety and reachability games as a criterion to choose between winning strategies and prove that one can compute a most permissive strategy (when it exists) in linear time.

Organisation. We illustrate by a relevant example the interest of our framework in Section 2. Section 3 gives the basic definitions and terminology used in this paper. Section 4 provides the translation to turn-based games and establishes the connections between strategies. Section 5 introduces the notion of permissiveness and shows how to compute most permissive strategies. Finally, we conclude in section 6.

2 A motivating example

A device driver is the interface between the hardware device and the application or the operating system. Being executed with supervisor privileges, any error in a driver may have a serious impact on the integrity of the entire system. A specific driver (as opposed to a generic driver) is a driver dedicated to an application, i.e., with a smaller memory footprint.

In the context of driver synthesis, the environment is both the hardware device and the application using the driver. Then uncontrollable actions are interrupts that are triggered by the hardware and the requests made by the application.

Let us consider an analog-to-digital converter (ADC) inspired by the one of the MPC5xx microcontrollers family. An ADC cell has multiplexed acquisition channels (only one channel at a time). In order to allow the conversion of several channels, the conversions are combined into a conversion chain.

There are two types of conversion: normal or injected. In a normal conversion, it is possible to make a chain of conversion uniquely (*oneShot*) or continuously (*scan*). In *oneShot* mode, the cell stops acquisition at the end of the conversion, while in *scan* mode it repeats the chain ad infinitum (until a stop action is performed).

If one wants to make a *oneShot* acquisition in the middle of a conversion in *scan* mode, it is possible to use the injected conversions. An injected conversion

is analogous to a software interrupt (`inject`) that can be maskable, i.e., it can be disabled. When an injected conversion is started, any conversion in progress is interrupted. The injected conversion is then carried out. At the end of the injected conversion, the chain which was interrupted resumes where it had been stopped (see Figure 1 for channel CH5). If a conversion is interrupted twice by two injected conversions, then it is lost and the *scan* goes to the next one (see Figure 1 for channel CH6).

Fig. 1: Conversion chain Scan for channels 3,5,6,8 with some injected conversions.

A conversion takes a non-null time that is not known precisely. At the end of a conversion the hardware generates an interrupt EOC (end of conversion). This interruption is ineluctable, i.e., it is guaranteed to happen eventually.

Finally, the converter can sleep or be awake but if a conversion request occurs while sleeping, the driver must return an error to the application. Let us take stock of what we need to model:

- controllable actions of the driver
- uncontrollable actions of the environment where:
  - some uncontrollable actions take a non-null time and cannot happen immediately (*oneShot*, *scan*, *eoc*);
  - some uncontrollable actions are guaranteed to happen eventually. The input state of such a transition is then a transient state (*oneShot*, *scan*, *eoc*);
  - some uncontrollable actions are *maskable* (`inject`): they can be disabled by the controller.

3 Definitions

The following definitions introduce a kind of concurrent game between the controller (denoted by C) and the environment (denoted by U). In all states $q \in Q$, C (resp. U) selects an action in $Avail_C(q) \subseteq A_C$ (resp. $Avail_U(q) \subseteq A_U$). As seen in Definition 2, it selects a qualitative delay for performing its action and it can block a subset of the maskable actions of the environment (A_U^m). C may also be inactive while the environment has to act in *transient states* (QT). Thus in all $q \in QT$, an unmaskable action is available, i.e. $Avail_U(q) \setminus A_U^m \neq \emptyset$.

Definition 1 (Game structure). *A* game structure *is a tuple* $\mathcal{G} = (Q, A_C, Avail_C, A_U, Avail_U, \delta)$ *where:*

- $Q = QT \uplus QI$ *is a set of states partitioned in transient states* QT *and idle states* QI *with* $q_0 \in Q$*, the initial state;*

– A_C *is the set of actions of the controller and* $Avail_C : Q \to 2^{A_C}$ *defines its available actions depending on states.*
– A_U *is the set of actions of the environment with* $A_C \cap A_U = \emptyset$ *and* $Avail_U : Q \to 2^{A_U}$ *defines its available actions depending on states.* A_U *includes the set of avoidable actions* A_U^a *and the set of maskable actions* A_U^m*.*
– $\delta : Q \times A_C \cup A_U \to Q$ *is the transition function such that* $\delta(q, a)$ *is defined if and only if* $a \in Avail_C(q) \cup Avail_U(q)$*. For all* $q \in QT$*,* $Avail_U(q) \setminus A_U^m \neq \emptyset$*.*

Example 1. The game structure of the case study presented in Section 2 is depicted in Figure 2. We use the following graphical notations: Idle (resp. transient) states states are represented by (resp. double) circles, controller (resp. environment) transitions are represented by solid (resp. dashed) arrows, avoidable transitions start with a small circle and maskable actions are written in `TrueType` font.

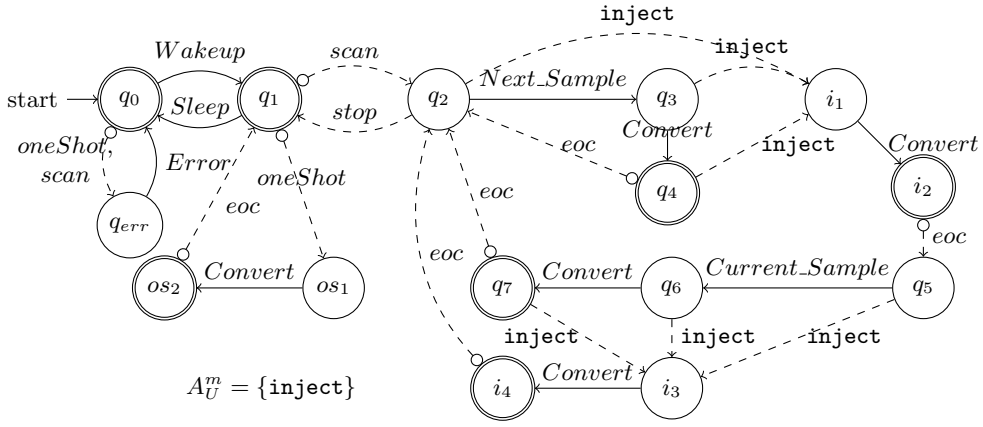

Fig. 2: Game structure of the case study

Given a current state q, the controller decides whether it intends to act ($\mathbf{0}$ or $\overline{\mathbf{0}}$) or not (ε), which action it intends to perform, when it will act (immediately: $\mathbf{0}$; or later: $\overline{\mathbf{0}}$) and which actions it will block.

Definition 2 (Decision). *Let* $\mathcal{G}$ *be a game structure and* $q \in Q$*. Then the set of* decisions *of the controller* $Dec(q)$ *is defined as follows.* $(\gamma, \tau, B) \in Dec(q)$ *if:*
– *the action* $\gamma \in Avail_C(q) \cup \{\varepsilon\}$ *where* ε *denotes* inaction*;*
– *the delay* $\tau \in \{\mathbf{0}, \overline{\mathbf{0}}, \varepsilon\}$ *fulfills* $\tau = \varepsilon$ *iff* $\gamma = \varepsilon$*;*
– *the actions to be masked* $B \subseteq A_U^m \cap Avail_U(q)$*.*

Given a state q and a decision d of the controller, the next state to be reached may be either (1) q itself if it is idle and the controller is inactive, either (2) the state reached by the action selected by the controller (if any), or (3) a state

reached by an environment action that has not be preempted by the action of the controller played without delay or masked by the controller.

Definition 3 (Play transitions). *Let $\mathcal{G}$ be a game structure, $q \in Q$ and $d = (\gamma, \tau, B) \in Dec(q)$, the set of* play transitions *$Next(q, d)$ is defined by:*

- *If $\gamma = \varepsilon$ and $q \in QI$ then $q \xrightarrow{d,\varepsilon} q \in Next(q, d)$;*
- *If $\gamma \neq \varepsilon$ then $q \xrightarrow{d,\gamma} \delta(q, \gamma) \in Next(q, d)$;*
- *For all $a \in Avail_U(q) \setminus (A_U^a \cup B)$, $q \xrightarrow{d,a} \delta(q, a) \in Next(q, d)$;*
- *If $\tau \neq \mathbf{0}$ then for all $a \in Avail_U(q) \cap A_U^a \setminus B$, $q \xrightarrow{d,a} \delta(q, a) \in Next(q, d)$.*

By construction, $Next(q, d)$ is never empty: if $\gamma \neq \varepsilon$ then $q \xrightarrow{d,\gamma} \delta(q, \gamma) \in Next(q, d)$ else if $q \in QI$ then $q \xrightarrow{d,\varepsilon} q \in Next(q, d)$ else there is some $a \in Avail_U(q) \setminus A_U^m$ such that $q \xrightarrow{d,a} \delta(q, a) \in Next(q, d)$.

A play is a finite or infinite sequence of play transitions such that the source of a non initial transition is the destination of the transition that precedes it.

Definition 4 (Play). *Let $\mathcal{G}$ be a game structure. Then $r = (q_n \xrightarrow{d_n,a_n} q_{n+1})_{n\in\mathbb{N}}$ where for all $n \in \mathbb{N}$, $d_n \in Dec(q_n)$ and $q_n \xrightarrow{d_n,a_n} q_{n+1} \in Next(q_n, d_n)$ is an* infinite play*. A* finite play *r is a finite prefix of an infinite play, ending in a state denoted* $\mathrm{Last}(r)$*. $\overline{\mathcal{R}}$ (resp. $\mathcal{R}$) denotes the set of infinite (resp. finite) plays. The empty play is denoted by λ with* $\mathrm{Last}(\lambda) = q_0$*.*

A strategy of the controller restricts the underlying transition system of the concurrent game by selecting a decision for all finite plays allowed by the strategy. Thus this mapping is inductively defined, simultaneously with its domain.

Definition 5 (Strategy). *A* controller strategy *s_C is a partial mapping from $\mathcal{R}$ to $\bigcup_{q\in Q} Dec(q)$ with its domain denoted $Dom(s_C)$ inductively defined by:*

- *$\lambda \in Dom(s_C)$;*
- *for all $r \in Dom(s_C)$, $s_C(r) \in Dec(\mathrm{Last}(r))$ and for all $\mathrm{Last}(r) \xrightarrow{s_C(r),a} q \in Next(\mathrm{Last}(r), s_C(r))$, $r(\mathrm{Last}(r) \xrightarrow{s_C(r),a} q) \in Dom(s_C)$.*

A play $r = (q_n \xrightarrow{d_n,a_n} q_{n+1})_{n\in\mathbb{N}}$ complies with *s_C if for all n, $d_n = s_C((q_m \xrightarrow{d_m,a_m} q_{m+1})_{m<n})$. The* outcome *of s_C, denoted by* $\mathrm{Outcome}(s_C)$*, is the set of infinite plays complying with it.*

Any decision of a *positional* (also called *memoryless*) strategy s_C only depends on the last state of the play. For such a strategy, given some $q \in Q$, $s_C(q)$ denotes the decision of s_C for any finite play with last state q.

A *goal* W for the controller is a subset of Q^ω. W.r.t. W, a strategy s_C is *winning* for q_0 if for all plays $(q_n \xrightarrow{d_n,a_n} q_{n+1})_{n\in\mathbb{N}}$ complying with s_C, $(q_n)_{n\in\mathbb{N}} \in$

W. A *parity goal* W_μ is defined by a mapping μ from Q to $\mathbb{N}$. Let $s = (q_n)_{n\in\mathbb{N}}$ define $m_\mu(s) = \max(i \mid \forall n\ \exists n' \geq n\ i = \mu(q_{n'}))$. Then $s \in W_\mu$ iff $m_\mu(s)$ is even. Parity goals include several kinds of goals like *safety* and *reachability* goals. A *game* is a pair $(\mathcal{G}, W)$.

4 From concurrent games to turn-based games

A turn-based game interpretation. In order to apply the theory and algorithms of parity turn-based games, we propose below a linear-time translation of a concurrent game structure $\mathcal{G}$ into a turn-based one $\widehat{\mathcal{G}}$ such that given some parity goal W: (1) the controller has a winning strategy for $(\mathcal{G}, W)$ iff it has a winning strategy $(\widehat{\mathcal{G}}, W)$ and (2) from a positional winning strategy in $(\widehat{\mathcal{G}}, W)$, one can build in linear time a positional winning strategy in $(\mathcal{G}, W)$.

Definition 6. *$\widehat{\mathcal{G}} = (\widehat{Q}, \to)$, a turn-based game structure is defined by:*
- *$\widehat{Q} = \widehat{Q}_C \uplus \widehat{Q}_U$, the set of states with $q_0 \in \widehat{Q}_C$, the initial state;*
- *$\to \subset \widehat{Q} \times \widehat{Q}$ the transition relation fulfilling: $\forall q \in \widehat{Q}\ \exists q' \in \widehat{Q}\ q \to q'$.*

We denote Own the mapping from $\widehat{Q}$ to $\{C, U\}$ defined for all $q \in \widehat{Q}$ by $Own(q) = C$ if and only if $q \in \widehat{Q}_C$. $(q_n)_{n\in\mathbb{N}} \in \widehat{Q}^\omega$ is an infinite *play* of $\widehat{\mathcal{G}}$ if for all $n \in \mathbb{N}$, $q_n \to q_{n+1}$. A sequence $(q_m)_{m\leq n} \in \widehat{Q}^*\widehat{Q}_C$ is a finite play if for all $m < n$, $q_m \to q_{m+1}$. Note that we only define finite plays ending in states owned by C, as only those will be useful to define strategies for C.

Definition 7 (Strategy). *A* strategy *s_C of $\widehat{\mathcal{G}}$ is a partial mapping from the set of of finite plays to $\widehat{Q}$ with its domain denoted $Dom(s_C)$ inductively defined by:*
- *$q_0 \in Dom(s_C)$;*
- *for all $\rho = (q_m)_{m\leq n} \in Dom(s_C)$, $s_C(\rho) \in \widehat{Q}$ with $q_n \to s_C(\rho)$ and for all $\rho'' = \rho s_C(\rho)\rho'$, such that $s_C(\rho)\rho' \in \widehat{Q}_U^*\widehat{Q}_C$ $\rho'' \in Dom(s_C)$.*

A play $\rho = (q_n)_{n\in\mathbb{N}}$ complies with *s_C if for all n, such that $q_n \in \widehat{Q}_C$, $q_{n+1} = s_C((q_m)_{m\leq n})$. The* outcome *of s_C, denoted* Outcome(s_C)*, is the set of infinite plays complying with it.*

Let $\widehat{W} \subseteq \widehat{Q}^\omega$ be a goal, s_C is winning in $(\widehat{\mathcal{G}}, \widehat{W})$, if Outcome$(s_C) \subseteq \widehat{W}$.

In order to obtain a canonical translation we assume an enumeration order of A_C and A_U. It should be be clear that the results hold whatever the chosen order. According to this order, define for all $q \in Q$:

- maskable uncontrollable actions:
  $Avail_U(q) \cap A_U^m = \{\alpha_1^q, \ldots, \alpha_{\ell_q}^q\}$;
- unavoidable maskable uncontrollable actions:
  $(Avail_U(q) \cap A_U^m) \setminus A_U^a = \{\alpha_1^q, \ldots, \alpha_{k_q}^q\}$ with $k_q \leq \ell_q$;
- unmaskable uncontrollable actions:
  $Avail_U(q) \setminus A_U^m = \{\beta_1^q, \ldots, \beta_{n_q}^q\}$;

– unavoidable unmaskable uncontrollable actions:
$Avail_U(q) \setminus (A_U^a \cup A_U^m) = \{\beta_1^q, \ldots, \beta_{m_q}^q\}$ with $m_q \leq n_q$;
– controllable actions:
$Avail_C(q) = \{\gamma_1^q, \ldots, \gamma_{p_q}^q\}$.

In order to avoid handling particular cases, Definition 8 assumes that for all q, $k_q \geq 1$, $n_q \geq 1$, and $p_q \geq 1$. Afterwards, we explain how to adapt the translation when this is not the case. Let us first informally describe how the turn-based version $\widehat{\mathcal{G}}$ of game $\mathcal{G}$ is specified:

– The states of $\mathcal{G}$ are also states of $\widehat{\mathcal{G}}$ and belong to the controller. In such a state q, the controller has three choices (see Figure 3):
  • either it decides to (try to) play immediately going to state $(q_{\mathbf{0}}^C, \alpha_1^q)$;
  • either it decides to (try to) play not immediately going to state $(q_{\overline{\mathbf{0}}}^C, \alpha_1^q)$;
  • or it decides to be inactive, going to state $(q_\varepsilon^C, \alpha_1^q)$.

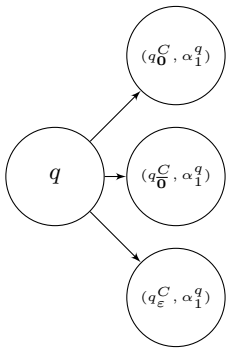

Fig. 3

– From state $(q_{\mathbf{0}}^C, \alpha_1^q)$, the controller successively either lets the environment the availability of action α_1^q by going to $(q_\varepsilon^U, \alpha_1^q)$ or masks this action by going to $(q_{\mathbf{0}}^C, \alpha_2^q)$. After all maskable unavoidable actions have been enumerated (and masked or not played), in $q_{\mathbf{0}}^U$ the environment can play any unavoidable and unmaskable action or, by going in q_0^C, let the controller play an action (see Figure 4).
– The situation from state $(q_{\overline{\mathbf{0}}}^C, \alpha_1^q)$ is similar to the previous one except that the controller first enumerates *all* the maskable actions and in $q_{\mathbf{0}}^U$ the environment can play *any* unmaskable action (see Figure 5).
– The situation from state $(q_\varepsilon^C, \alpha_1^q)$ is similar to the previous one except that in q_ε^U, if q is transient the environment must play *some* unmaskable action while if q is idle it can decide to be inactive going back to q (see Figure 6).

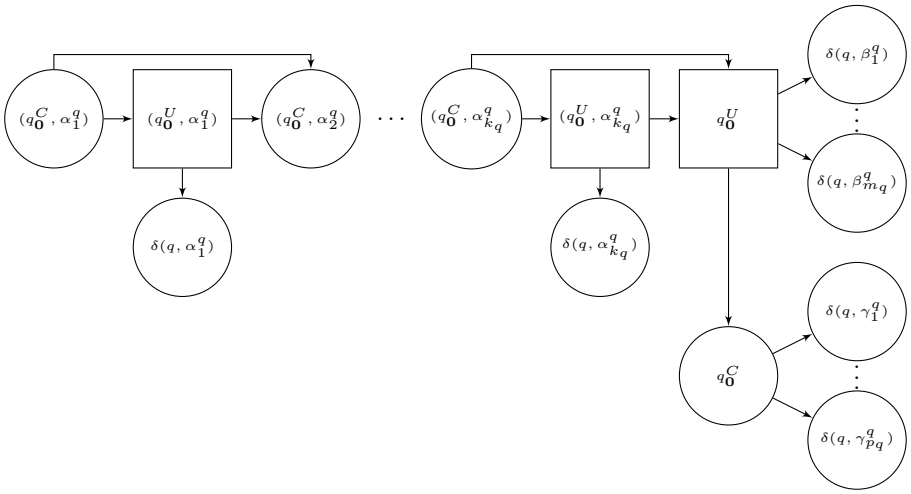

Fig. 4

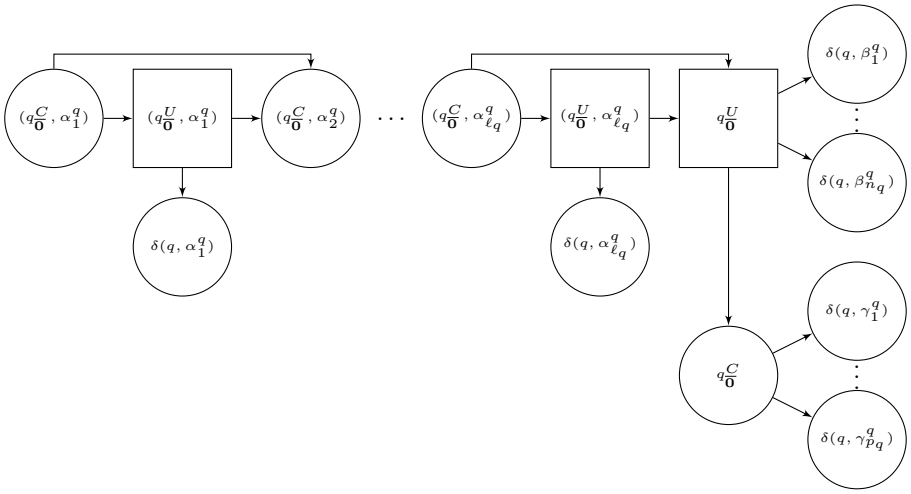

Fig. 5

Definition 8. *Let $\mathcal{G}$ be a game structure. Then $\widehat{\mathcal{G}} = (\widehat{Q}, Own, \rightarrow)$, a turn-based game structure, is defined as follows:*

- $\widehat{Q} = Q \cup \{q^y_x \mid q \in Q \wedge ((x \in \{\mathbf{0}, \overline{\mathbf{0}}\} \wedge y \in \{C, U\}) \vee (x = \varepsilon \wedge y = U))\}$ $\cup \{(q^y_x, \alpha^q_i)) \mid q \in Q \wedge y \in \{C, U\} \wedge (x = \mathbf{0} \wedge (i \leq k_q)) \vee (x \in \{\overline{\mathbf{0}}, \varepsilon\} \wedge (i \leq \ell_q))\}$;
- $\widehat{Q}_C = Q \cup \{q^C_x\}_{q,x} \cup \{(q^C_x, z)\}_{q,x,z}$, $\widehat{Q}_U = Q \cup \{q^U_x\}_{q,x} \cup \{(q^U_x, z)\}_{q,x,z}$;
- $\rightarrow$ *is defined by for all $q \in Q$ and all $x \in \{\mathbf{0}, \overline{\mathbf{0}}, \varepsilon\}$, $q \rightarrow (q^C_x, \alpha^q_1)$ and:*
  Case of immediate play controller choice.
  - *For all q and $i \leq k_q$, $(q^C_{\mathbf{0}}, \alpha^q_i) \rightarrow (q^U_{\mathbf{0}}, \alpha^q_i)$ and $(q^U_{\mathbf{0}}, \alpha^q_i) \rightarrow \delta(q, \alpha^q_i)$;*

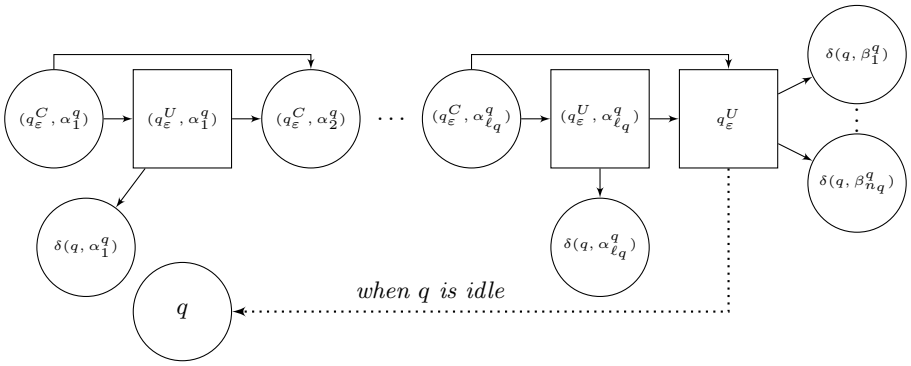

Fig. 6

- *For all* q *and* $i < k_q$, $(q_{\mathbf{0}}^C, \alpha_i^q) \to (q_{\mathbf{0}}^C, \alpha_{i+1}^q)$ *and* $(q_{\mathbf{0}}^U, \alpha_i^q) \to (q_{\mathbf{0}}^C, \alpha_{i+1}^q)$;
- *For all* q, $(q_{\mathbf{0}}^C, \alpha_{k_q}^q) \to q_{\mathbf{0}}^U$ *and* $(q_{\mathbf{0}}^U, \alpha_{k_q}^q) \to q_{\mathbf{0}}^U$;
- *For all* q *and* $i \leq m_q$, $q_{\mathbf{0}}^U \to \delta(q, \beta_i^q)$ *and* $q_{\mathbf{0}}^U \to q_{\mathbf{0}}^C$;
- *For all* q *and* $i \leq p_q$, $q_{\mathbf{0}}^C \to \delta(q, \gamma_i^q)$.

Other Cases ($x \in \{\overline{\mathbf{0}}, \varepsilon\}$)**.**

- *For all* q *and* $i \leq \ell_q$, $(q_x^C, \alpha_i^q) \to (q_x^U, \alpha_i^q)$ *and* $(q_x^U, \alpha_i^q) \to \delta(q, \alpha_i^q)$;
- *For all* q *and* $i < \ell_q$, $(q_x^C, \alpha_i^q) \to (q_x^C, \alpha_{i+1}^q)$ *and* $(q_x^U, \alpha_i^q) \to (q_x^C, \alpha_{i+1}^q)$;
- *For all* q, $(q_x^C, \alpha_{\ell_q}^q) \to q_x^U$ *and* $(q_x^U, \alpha_{\ell_q}^q) \to q_x^U$;
- *For all* q *and* $i \leq n_q$, $q_x^U \to \delta(q, \beta_i^q)$;
- $q_{\overline{\mathbf{0}}}^U \to q_{\overline{\mathbf{0}}}^C$ *and for all* $q \in Q$ *and* $i \leq p_q$, $q_{\overline{\mathbf{0}}}^C \to \delta(q, \gamma_i^q)$;
- *When* $q \in QI$, $q_\varepsilon^U \to q$.

Let us explain how to address the particular cases. For instance when ℓ_q=0, the three transitions outgoing from q target the states $q_{\mathbf{0}}^U$, $q_{\overline{\mathbf{0}}}^U$ and q_ε^U. The other particular cases are similarly handled.

Example 2. Figure 7 partly illustrates this translation for the game structure of Figure 2, starting from transient state q_0. In all figures illustrating $\widehat{\mathcal{G}}$, states owned by the controller are represented by circles while states owned by the environment are represented by rectangles.

Correspondence between plays. We want to establish a correspondence between plays of $\mathcal{G}$ and plays of $\widehat{\mathcal{G}}$. With this aim, we introduce *connecting paths* of $\widehat{\mathcal{G}}$.

Definition 9. *A* connecting path *of* $\widehat{\mathcal{G}}$ *is a path from some* $q \in Q$ *to some* $q' \in Q$ *that does not visit in between other states of* Q.

Let ρ be a connecting path. We denote $src(\rho)$ the source state of ρ and $dst(\rho)$ its destination state. Given two connecting paths ρ, ρ' such that q, the target state of ρ, is the source state of ρ', $\rho\rho'$ denotes the concatenation of ρ and ρ' without the repetition of q. As usual, this operation is extended to finite or

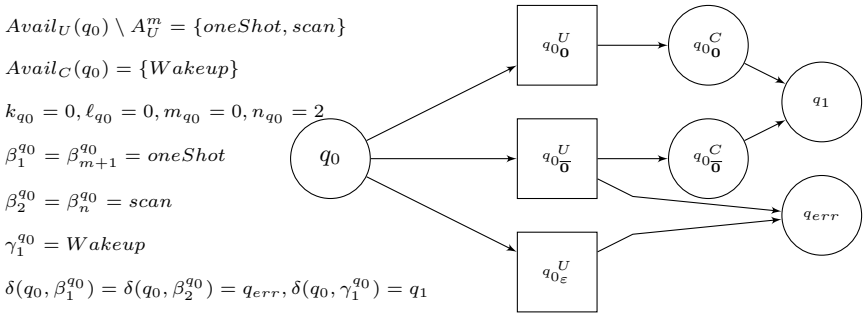

Fig. 7: The controller choices from q_0

countable paths. Observe that any infinite play of $\widehat{\mathcal{G}}$ is a countable concatenation of connecting paths.

The following lemma lists all possible connecting paths. We omit its proof as it straightforwardly comes from examination of the local structure of $\widehat{\mathcal{G}}$. Observe that we associate an action with a connecting path.

Lemma 1. *Let $q \in Q$. Then all connecting paths starting from q can be concisely specified (with an associated action a_ρ) as follows:*

- *$\rho(q, y, B, \alpha_i^q)$, the path that (1) starts from q to (q_y^C, α_1^q), (2) goes to (q_y^U, α_i^q) avoiding the set $\{(q_y^U, \alpha_j^q) \mid \alpha_j^q \in B\}$, and (3) reaches $\delta(q, \alpha_i^q)$. $a_\rho = \alpha_i^q$;*
- *$\rho(q, y, B, \beta_i^q)$, the path that (1) starts from q to (q_y^C, α_1^q), (2) goes to q_y^U avoiding the set $\{(q_y^U, \alpha_j^q) \mid \alpha_j^q \in B\}$, and (3) reaches $\delta(q, \beta_i^q)$. $a_\rho = \beta_i^q$;*
- *$\rho(q, y, B, \gamma_i^q)$, the path that (1) starts from q to (q_y^C, α_1^q), (2) goes to q_y^C avoiding the set $\{(q_y^U, \alpha_j^q) \mid \alpha_j^q \in B\}$, and (3) reaches $\delta(q, \gamma_i^q)$. $a_\rho = \gamma_i^q$;*
- *when q is idle, $\rho(q, \varepsilon, B, \varepsilon)$, the path that (1) starts from q to $(q_\varepsilon^C, \alpha_1^q)$, (2) goes to q_ε^U avoiding the set $\{(q_y^U, \alpha_j^q) \mid \alpha_j^q \in B\}$, and (3) returns to q. $a_\rho = \varepsilon$.*

We now relate play transitions of $\mathcal{G}$ to connecting paths of $\widehat{\mathcal{G}}$.

Definition 10. *Let $e = q \xrightarrow{d,a} q'$ with $d = (\gamma, \tau, B)$ be a play transition of $\mathcal{G}$. Then the connecting path $\hat{e}$ is defined as follows:*

- *If $a = \alpha_i^q$ for some α_i^q then $\hat{e} = \rho(q, \tau, B', \alpha_i^q)$ with $B' = \{\alpha_j^q \mid j < i\} \cap B$;*
- *If $a = \beta_i^q$ for some β_i^q then $\hat{e} = \rho(q, \tau, B, \beta_i^q)$*
- *If $a = \gamma_i^q$ for some γ_i^q then $\hat{e} = \rho(q, \tau, B, \gamma_i^q)$;*
- *If $a = \varepsilon$ then $\hat{e} = \rho(q, \tau, B, \varepsilon)$;*

We now extend in a natural way the correspondence between transitions to plays.

Definition 11 (Relation between plays). *Let $r = (e_n)_{n \leq N}$ (resp. $r = (e_n)_{n \in \mathbb{N}}$) be a finite (resp. infinite) play of $\mathcal{G}$. Then $\hat{r}$ a finite (resp. infinite) play of $\widehat{\mathcal{G}}$, is defined by $\hat{r} = (\hat{e}_n)_{n \leq N}$ (resp. $\hat{r} = (\hat{e}_n)_{n \in \mathbb{N}}$).*

Correspondence between strategies. Let $\mathcal{G}$ be a concurrent game structure and $\rho = (q_n)_{n\in\mathbb{N}} \in \widehat{Q}^\omega$. Define $\pi(\rho)$ as $(q_{\alpha(n)})_{n\in\mathbb{N}}$ where α is a strictly increasing mapping from $\mathbb{N}$ to $\mathbb{N}$ with the range of α being $\{n \mid q_n \in Q\}$. This means, π extracts the subsequence of states from the original game from any play in the turn-based game. Let W be a goal of $\mathcal{G}$. Then $\widehat{W} \subseteq \widehat{Q}^\omega$, a goal of $\widehat{\mathcal{G}}$, is defined by $\widehat{W} = \{\rho \mid \pi(\rho) \in W\}$.

In order to obtain a correspondence between strategies, we focus on translating a decision $d \in Dec(q)$ in $\mathcal{G}$ into a *local positional strategy* $\hat{d}$ of $\widehat{\mathcal{G}}$. Given a state q and a decision $d = (\gamma, \tau, B)$ induced by a strategy in $\mathcal{G}$, the 'local' corresponding strategy in $\widehat{\mathcal{G}}$ consists in allowing exactly the connecting paths $\hat{e}$ such that $e \in Next(q, d)$. Thus, (1) in q the strategy selects (q^C_τ, α^q_1), (2) in all states (q^C_τ, α^q_i), it selects (q^U_τ, α^q_i) if $\alpha^q_i \notin B$ and avoids it otherwise, and (3) when $\gamma \neq \varepsilon$, it selects in q^C_τ the state $\delta(q, \gamma)$.

Definition 12 (From decisions to local strategies). *Let $q \in Q$ and $d = (\gamma, \tau, B) \in Dec(q)$. The partial mapping $\hat{d} : \hat{Q} \to \hat{Q}$ is defined as follows:*

- $\hat{d}(q) = (q^C_\tau, \alpha^q_1)$;
- *for all (defined)* (q^C_τ, α^q_i) *if* $\alpha^q_i \notin B$ *then* $\hat{d}(q^C_\tau, \alpha^q_i) = (q^U_\tau, \alpha^q_i)$;
- *for all (defined)* (q^C_τ, α^q_i) *if* $\alpha^q_i \in B$ *and* $(i, \tau) \notin \{(k_q, \mathbf{0}), (\ell_q, \overline{\mathbf{0}}), (\ell_q, \varepsilon)\}$ *then* $\hat{d}(q^C_\tau, \alpha^q_i) = (q^C_\tau, \alpha^q_{i+1})$;
- *for all (defined)* (q^C_τ, α^q_i) *if* $\alpha^q_i \in B$ *and* $(i, \tau) \in \{(k_q, \mathbf{0}), (\ell_q, \overline{\mathbf{0}}), (\ell_q, \varepsilon)\}$ *then* $\hat{d}(q^C_\tau, \alpha^q_i) = q^U_\tau$;
- *If* $\gamma \neq \varepsilon$ *then* $\hat{d}(q^C_\tau)) = \delta(q, \gamma)$.

Definition 13. *Let $d \in Dec(q)$. A connecting path ρ starting from q* complies with *d if for all transitions $q_1 \to q_2$ of ρ with $q_1 \in \widehat{Q}_C$, $q_2 = \hat{d}(q_1)$.*

By examining all possible cases, one gets the following lemma.

Lemma 2. *Let $q \in Q$ and $d = (\gamma, \tau, B) \in Dec(q)$. Then:*

$$Next(q, d) = \{q \xrightarrow{d, a_\rho} dst(\rho) \mid \rho \text{ is a connecting path with } src(\rho) = q \text{ complying with } d\}$$

Example 3. Let us consider the example of Figure 2. Figure 8 shows the result of the decision $d = (\gamma^{q_2}_1, \mathbf{0}, \{\alpha^{q_2}_1\}) = (Next\_Sample, \mathbf{0}, \{\texttt{inject}\})$ from q_2.

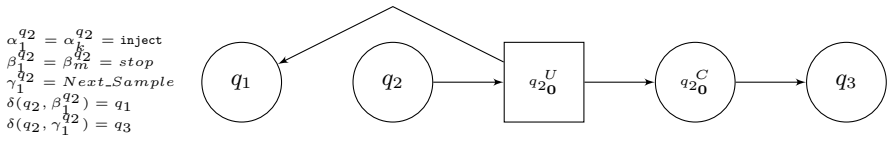

Fig. 8: The connecting paths from q_2 complying with $d = (Next\_Sample, \mathbf{0}, \{\texttt{inject}\})$

Given the decisions selected by a strategy s_C of $\mathcal{G}$ and applying the corresponding local strategies, one gets a strategy $\hat{s}_C$ of $\widehat{\mathcal{G}}$. The next definition is

sound since, by induction and using Lemma 2, for every finite play ρ that complies with $\hat{s}_C$ and ends in Q, there is some r_ρ that complies with s_C such that $\rho = \hat{r}$.

Let $q \in Q$, $\tau \in \{\mathbf{0}, \overline{\mathbf{0}}, \varepsilon\}$, $\alpha \in A_U^m$ such that $(q_\tau^U, \alpha) \in \widehat{Q}$ and $B \subseteq A_U^m \cap Avail_U(q)$. Then $(q_\tau^U, \alpha)_{\downarrow B}$ denotes (q_τ^U, α) if $\alpha \notin B$ and the empty word of $\widehat{Q}^*$ otherwise.

Definition 14. *Let s_C be a controller strategy of $\mathcal{G}$. Then $\hat{s}_C$, a controller strategy of $\widehat{\mathcal{G}}$ is defined by induction on the length of $\rho = \hat{r_\rho}$ where r_ρ is a finite play of $\mathcal{G}$ complying with s_C and ending in some $q \in Q$ as follows when denoting $s_C(r)$ by $d = (\gamma, \tau, B)$.*

- *$\hat{s}_C(\rho) = \hat{d}(q)$;*
- *for all (defined) (q_τ^C, α_i^q), $\hat{s}_C(\rho(q_\tau^C, \alpha_1^q)(q_\tau^U, \alpha_1^q)_{\downarrow B} \dots (q_\tau^C, \alpha_i^q)) = \hat{d}(q_\tau^C, \alpha_i^q)$*
- *If $\gamma \neq \varepsilon$ then $\hat{s}_C(\rho(q_\tau^C, \alpha_1^q)(q_\tau^U, \alpha_1^q)_{\downarrow B} \dots (q_\tau^C, \alpha_{k_q}^q)(q_\tau^U, \alpha_{k_q}^q)_{\downarrow B} q_\tau^U q_\tau^C)) = \hat{d}(q_\tau^C))$.*

Let $\rho^* = \rho\rho'$ where ρ' is a a connecting path complying with d and associated action $a_{\rho'}$. Then $r_{\rho^*} = r_\rho(\mathrm{Last}(r_\rho) \xrightarrow{d, a_{\rho'}} dst(\rho'))$.

Applying inductively Lemma 2, one immediately gets the next proposition and corollary:

Proposition 1. *Let s_C be a controller strategy of $\mathcal{G}$. Then:*

$$\mathrm{Outcome}(\hat{s}_C) = \{\hat{r} \mid r \in \mathrm{Outcome}(s_C)\}$$

Corollary 1. *Let $Goal \subseteq Q^\omega$ be a goal and s_C be a winning strategy for $Goal$ in $\mathcal{G}$. Then $\hat{s}_C$ is a winning strategy for $Goal$ in $\widehat{\mathcal{G}}$.*

In order to get the converse result we exploit the fact that positional strategies are sufficient for turn-based parity games.

Definition 15. *Let s_C be a positional controller strategy of $\widehat{\mathcal{G}}$. Then $\overline{s}_C$ is a positional controller strategy of $\mathcal{G}$ defined as follows. Let $q \in Q$ and denote $\overline{s}_C(q) = (\gamma, \tau, B)$.*

- *If $s_C(q) = (q_{\mathbf{0}}^C, \alpha_1^q)$ and $s_C(q_{\mathbf{0}}^C) = \delta(a, \gamma_i^q)$ then $\gamma = \gamma_i^q$, $\tau = \mathbf{0}$ and $B = \{i \leq k_q \mid s_C((q_{\mathbf{0}}^C, \alpha_i^q)) \neq (q_{\mathbf{0}}^U, \alpha_i^q)\}$;*
- *If $s_C(q) = (q_{\overline{\mathbf{0}}}^C, \alpha_1^q)$ and $s_C(q_{\overline{\mathbf{0}}}^C) = \delta(a, \gamma_i^q)$ then $\gamma = \gamma_i^q$, $\tau = \overline{\mathbf{0}}$ and $B = \{i \leq \ell_q \mid s_C((q_{\overline{\mathbf{0}}}^C, \alpha_i^q)) \neq (q_{\overline{\mathbf{0}}}^U, \alpha_i^q)\}$;*
- *If $s_C(q) = (q_\varepsilon^C, \alpha_1^q)$ then $\gamma = \varepsilon$, $\tau = \varepsilon$ and $B = \{i \leq \ell_q \mid s_C((q_{\mathbf{0}}^C, \alpha_i^q)) \neq (q_{\mathbf{0}}^U, \alpha_i^q)\}$;*

Observe that $\overline{s}_C(q)$ is an item of $Dec(q)$, say d, and that s_C restricted to the subset of states of $\widehat{Q}$ related to q can be viewed as a local strategy, say d'. By construction, $d' = \hat{d}$. Thus applying inductively Lemma 2, one immediately gets the next proposition and corollary:

Proposition 2. *Let s_C be a positional controller strategy of $\widehat{\mathcal{G}}$. Then:*

$$\mathrm{Outcome}(s_C) = \{\hat{r} \mid r \in \mathrm{Outcome}(\overline{s}_C)\}.$$

Corollary 2. *Let $Goal \subset Q^\omega$ be a parity goal and s_C be a winning positional strategy for $Goal$ in $\widehat{\mathcal{G}}$. Then $\overline{s}_C$ is a winning positional strategy for $Goal$ in $\mathcal{G}$.*

Combining corollaries 1 and 2, one gets:

Theorem 1. *Let $\mathcal{G}$ be a concurrent game, $Goal \subseteq Q^\omega$ be a parity goal.*
- *Let s_C be a controller strategy. Then s_C is a winning strategy for $Goal$ if and only if $\hat{s}_C$ is a winning strategy for $Goal$;*
- *If there is a winning strategy for $\mathcal{G}$, there is a positional one, i.e. $\overline{s}'_C$ where s'_C is any positional winning strategy of $\widehat{\mathcal{G}}$.*

5 Permissivity of the controller

Permissivity is a criterion that has to be taken into account for choosing between possible controllers. In our context and due to the results of the previous section, we limit ourselves to positional controllers. First we introduce an order between controller decisions. In words, a decision d' is more permissive than d if (1) either d' is inactive or intends to play the same action as d, (2) the delay before acting of d' is greater or equal than the delay before acting of d, and (3) the set of actions masked by d' is a subset of actions masked by d. Here permissivity should be interpreted as the controller avoiding to restrict the behaviour of the environment.

Definition 16. *Let q be a state and $d = (\gamma, \tau, B)$ and $d' = (\gamma', \tau, B)$ be two decisions, one says that d' is more (or equally) permissive than d denoted by $d \preceq d'$ if: $\gamma' = \gamma$ or $\gamma' = \varepsilon$, $\tau \leq \tau'$ and $B' \subseteq B$.*

Assuming that there exists a winning strategy for q_0, our goal is to define and synthesise a maximally permissive winning strategy for q_0 w.r.t. safety and reachability goals. As explained later we introduce slightly different notions of maximally permissive winning strategy depending on the kind of goal. Observe that in the next definition, we do not care about losing states of the game since winning strategies do not enter losing states.

Definition 17. *Let s_C and s'_C be positional winning controller strategies of $(\mathcal{G}, Goal)$ where $Goal$ is a safety goal. Then s'_C is* more permissive *than s_C w.r.t. $Goal$, denoted $s_C \preceq s'_C$, if for all q winning state of $(\mathcal{G}, Goal)$, $s_C(q) \preceq s'_C(q)$. Additionally, $s_C \prec s'_C$ if $s_C \preceq s'_C$ and $s'_C \not\preceq s_C$.*
s_C is a maximally permissive *winning strategy w.r.t. $Goal$ if there is no s'_C such that $s_C \prec s'_C$.*

The synthesis of a maximally permissive winning strategy w.r.t. a safety goal is easy. Since it is enough to stay in the winning states, maximally permissive decisions can be combined without any restriction to get a maximally permissive winning strategy.

Theorem 2. *Let $\mathcal{G}$ be a concurrent game and $Goal$ be a safety goal. Once the winning states of $(\widehat{\mathcal{G}}, Goal)$ have been computed, a maximally permissive winning strategy of $(\mathcal{G}, Goal)$ for q_0 can be computed in linear time.*

Proof. We compute for all q, winning state of $(\mathcal{G}, Goal)$ (and thus of $(\widehat{\mathcal{G}}, Goal)$), a most permissive decision. Since we deal with a safety goal, any combination of decisions works. A maximally permissive decision for q is computed as follows.

- If $(q^C_\varepsilon, \alpha^q_1)$ is winning then define $B = \{\alpha^q_i \mid i \leq \ell_q \wedge \delta(q, \alpha^q_i) \text{ is not winning}\}$. By construction, $(\varepsilon, \varepsilon, B)$ is a maximally permissive decision;
- If $(q^C_\varepsilon, \alpha^q_1)$ is not winning and $(q^C_{\mathbf{0}}, \alpha^q_1)$ is winning then define $B = \{\alpha^q_i \mid i \leq \ell_q \wedge \delta(q, \alpha^q_i) \text{ is not winning}\}$. The set $\{\gamma^q_i \mid i \leq p_q \wedge \delta(q, \gamma^q_i) \text{ is winning}\}$ is not empty since $(q^C_{\mathbf{0}}, \alpha^q_1)$ is winning. Pick some γ^q_i inside. By construction, $(\gamma^q_i, \overline{\mathbf{0}}, B)$ is a maximally permissive decision;
- If $(q^C_\varepsilon, \alpha^q_1)$ and $(q^C_{\mathbf{0}}, \alpha^q_1)$ are not winning then $(q^C_{\mathbf{0}}, \alpha^q_1)$ is winning. Define $B = \{\alpha^q_i \mid i \leq k_q \wedge \delta(q, \alpha^q_i) \text{ is not winning}\}$. The set $\{\gamma^q_i \mid i \leq p_q \wedge \delta(q, \gamma^q_i) \text{ is winning}\}$ is not empty since $(q^C_{\mathbf{0}}, \alpha^q_1)$ is winning. Pick some γ^q_i inside. By construction, $(\gamma^q_i, \mathbf{0}, B)$ is a maximally permissive decision.

□

The definition of a maximally permissive winning strategy w.r.t. a reachability goal is more involved since we want to take into account the *delay* before reaching the target state q_f assumed w.l.o.g. to be absorbing (i.e., its only outgoing transition is a self-loop).

Definition 18. *Let $\mathcal{G}$ be a concurrent game structure and Goal be a reachability goal defined by an absorbing target state q_f. Let s_C be a positional winning strategy and $q \in Q$ be a winning state. Then $delay_{s_C}(q)$, the delay of $q \neq q_f$ w.r.t. s_C is defined by:*

$$delay_{s_C}(q) = \max(n \mid (q_m \xrightarrow{d_m, a_m} q_{m+1})_{1 \leq m < n} \textit{complying to } s_C, q = q_1, \forall i\ q_i \neq q_f)$$

By convention, $delay_{s_C}(q_f) = 0$.
The minimal delay *of a q, denoted $delay^*(q)$ is defined by:*

$$delay^*(q) = \min(delay_{s_C}(q) \mid s_C \textit{ is a winning strategy})$$

We also restrict the constraint for maximality for decisions of a strategy s_C to states *visited* by s_C: a state q is visited by s_C if there is a finite play complying to s_C starting in q_0 and ending in q.

Definition 19. *Let s_C and s'_C be positional winning controller strategies of $(\mathcal{G}, Goal)$ where Goal is a reachability goal defined by an absorbing target state q_f. Then s'_C is* more permissive *than s_C w.r.t. Goal, denoted $s_C \preceq s'_C$, if for all q state visited by s_C, $s_C(q) \preceq s'_C(q)$ and $delay_{s'_C}(q) \leq delay_{s_C}(q)$.*
$s_C \prec s'_C$ if $s_C \preceq s'_C$ and $s'_C \not\preceq s_C$.
s_C is a maximally permissive *winning strategy w.r.t. Goal if there is no s'_C such that $s_C \prec s'_C$.*

Let us informally explain how the proof of the next theorem proceeds to synthesise a maximally permissive winning strategy. If $q_0 = q_f$ we are done. Otherwise define $Q_0 = \{q_f\}$ and Q_1 to be the set of states $q \neq q_f$ for which there is a decision d such that all the transitions of $Next(q, d)$ lead to Q_0. Observe that

this set is non empty. Otherwise q_0 could not be a winning state. For such q select d maximally permissive among these kinds of decisions. If $q_0 \in Q_1$ we are done. We iterate this process until q_0 is found. Assume $Q_0, \ldots, Q_i$ have been built with $q_0 \notin \biguplus_{j \leq i} Q_j$. Define Q_{i+1} to be the set of states $q \notin \biguplus_{j \leq i} Q_j$ for which there is decision d such that all the transitions of $Next(q,d)$ lead to $\biguplus_{j \leq i} Q_j$. Observe that this set is non empty. Otherwise q_0 could not be a winning state. For such q select d maximally permissive among these kinds of decisions. Since Q is finite this process must stop with q_0 belonging to some Q_{i^*}.

Theorem 3. *Let $\mathcal{G}$ be a concurrent game structure and Goal be a reachability goal defined by an absorbing target state q_f. Then a maximally permissive winning strategy of $(\mathcal{G}, Goal)$ for q_0 can be computed in linear time.*

Proof. As explained we proceed by iteratively building disjoint sets of states $Q_0, Q_1, \ldots$ as long as q_0 does not belong to these states. Furthermore we associate decisions with every state belonging to these sets, that define the strategy s_C. We set $Q_0 = \{q_f\}$ and since q_f is absorbing, the associated decision is $(\varepsilon, \varepsilon, \emptyset)$. Assume that $Q_0, Q_1, \ldots, Q_i$ have been built with $q_0 \notin \biguplus_{j \leq i} Q_j$. For sake of readability, we write $Q' = \biguplus_{j \leq i} Q_j$. Let $q \notin Q'$.

- If q is transient and for all $k \leq n_q, \delta(q, \beta_k^q) \in Q'$ then $q \in Q_{i+1}$ and defining $B = \{\alpha_k^q \mid k \leq \ell_q \wedge \delta(q, \alpha_k^q) \notin Q'\}$, $(\varepsilon, \varepsilon, B)$ is the decision associated with q;
- Else if for all $k \leq n_q, \delta(q, \beta_k^q) \in Q'$ and there exists γ_p^q such that $\delta(q, \gamma_p^q) \in Q'$ then $q \in Q_{i+1}$ and defining $B = \{\alpha_k^q \mid k \leq \ell_q \wedge \delta(q, \alpha_k^q) \notin Q'\}$, $(\gamma_p^q, \overline{\mathbf{0}}, B)$ is the decision associated with q;
- Else if for all $k \leq m_q, \delta(q, \beta_k^q) \in Q'$ and there exists γ_p^q such that $\delta(q, \gamma_p^q) \in Q'$ then $q \in Q_{i+1}$ and defining $B = \{\alpha_k^q \mid k \leq k_q \wedge \delta(q, \alpha_k^q) \notin Q'\}$, $(\gamma_p^q, \mathbf{0}, B)$ is the decision associated with q.

By construction, for all $q \in Q$, one has $q \in Q_i$ if and only if $delay^*(q) = i$. Furthermore $delay_{s_C}(q) = delay^*(q)$. Assume there exists a winning strategy s'_C such that $s_C \prec s'_C$. Since s_C achieves the minimality of delay for states visited by s_C, there must exist a state q visited by s_C that belongs to some Q_i such that $s'_C(q)$ is strictly more permissive than $s_C(q)$. By definition of $s_C(q)$, this implies that there exists a state $q'' \notin \biguplus_{j<i} Q_j$ which is the destination of a transition of $Next(q,d)$. Thus $delay_{s'_C}(q) > delay_{s_C}(q)$, establishing a contradiction.
In order to obtain a linear time algorithm, one proceeds as follows. In the sequel i denotes the current iteration when the sets $Q_0, Q_1, \ldots, Q_{i-1}$ have been built. Here Q' denotes $Q' = \biguplus_{j \leq i} Q_j$.

- Initially one builds a "reverse graph" whose vertices are the states and there is an edge $q' \xrightarrow{\beta_k^q} q$ (resp. $q' \xrightarrow{\gamma_p^q} q$) with $k \leq m_q$ (resp. $p \leq p_q$) if $\delta(q, \beta_k^q) = q'$ (resp. $\delta(q, \gamma_p^q) = q'$).
- At beginning of the i^{th} iteration, the set Q_i has already been computed using the variables described below.
- One associates a counter c_q and a boolean b_q with every state q and a boolean b_q with every idle state q. Boolean b_q is true if there exists some action γ_p^q such that $\delta(q, \gamma_p^q) \in Q'$ and c_q is the size of the set $\{k \leq m_q, \delta(q, \beta_k^q) \notin Q'\}$.

A state q belongs to Q_i if (1) it does not belong to Q', (2) its counter c_q is null, and (3) either it is transient or its boolean b_q is true.
- The i^{th} iteration consists in two stages for all $q \in Q_i$. First one determines $s_C(q)$ using the rules described above. Then one updates the counters and the booleans using the edges of the reverse graph: for all $q \xrightarrow{\beta_k^{q'}} q'$, $c_{q'}$ is decremented and if there exists some $q \xrightarrow{\gamma_p^{q'}} q'$ $b_{q'}$ is set to true. If q' satisfies the three conditions w.r.t. iteration $i+1$, it enters the set Q_{i+1}.

It is routine to check that this procedures operates in linear time. □

6 Conclusion

We have introduced a model of qualitative time concurrent game between a controller and an environment. We have designed a linear-time translation from such a game to an untimed turn-based game and shown that given any parity goal, the concurrent game is winning for the controller if and only if the turn-based game is winning for it. This allows us, taking as input a positional winning strategy of the turn-based game, to build in linear time a positional winning strategy in the original game. Furthermore we have introduced a notion of permissivity for strategies and we have established that one can compute in linear time a maximally permissive winning strategy for safety and reachability goals. For future work, we want to design an algorithm for computing in polynomial time a maximally permissive winning strategy for repeated reachability goals. We also plan to modify the notion of delay for reachability goals by integrating the delay of actions when computing a maximally permissive winning strategy for reachability. Finally our notion of qualitative delay for actions could be refined in order to take into account more precise (but still qualitative) information.

References

1. P. Ashok, J. Křetínský, K. G. Larsen, A. Le Coënt, J. H. Taankvist, and M. Weininger. SOS: Safe, optimal and small strategies for hybrid Markov decision processes. In *QEST 2019*, volume 11785 of *LNCS*, pages 147–164, 2019.
2. J.-L. Béchennec, D. Lime, and O. H. Roux. Control of DES with urgency, avoidability and ineluctability. In J. Keller and W. Penczek, editors, *ACSD 2019*, pages 92–101, Aachen, Germany, 2019. IEEE Computer Society.
3. J.-L. Béchennec, D. Lime, and O. H. Roux. Logical Time Control of concurrent DES. *Discrete Event Dynamic Systems - Theory and Applications (DEDS)*, Jan. 2021. Springer.
4. G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and D. Lime. UPPAAL-Tiga: Time for playing games! In *CAV 2007*, volume 4590 of *LNCS*, pages 121–125, 2007.
5. K. Chatterjee, L. de Alfaro, and T. A. Henzinger. Strategy improvement for concurrent reachability and safety games. http://arxiv.org/abs/1201.2834, 2012.

6. L. De Alfaro, M. Faella, T. A. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *CONCUR 2003*, volume 2761 of *LNCS*, pages 144–158, 2003.
7. L. de Alfaro, T. A. Henzinger, and O. Kupferman. Concurrent reachability games. *Theoretical Computer Science*, 386(3):188 – 217, 2007.
8. L. De Alfaro, T. A. Henzinger, and R. Majumdar. Symbolic algorithms for infinite-state games. In *CONCUR 2001*, volume 2154 of *LNCS*, pages 536–550, 2001.
9. C. Golaszewski and P. Ramadge. Control of discrete event processes with forced events. In *Proceedings of the 26th Conference on Decision and Control*, pages 247–251, 1987.
10. M. Jurdzinski and A. Trivedi. Reachability-time games on timed automata. In *ICALP'07*, volume 4596 of *LNCS*, pages 838–849, 2007.
11. O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *STACS 95*, volume 900 of *LNCS*, pages 229–242, 1995.
12. P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control Optim.*, 25(1):206–230, Jan. 1987.
13. W. Thomas. On the synthesis of strategies in infinite games. In *STACS 95*, volume 900 of *LNCS*, pages 1–13, 1995.
14. W. M. Wonham and P. J. Ramadge. On the supremal controllable sublanguage of a given language. In *The 23rd IEEE Conference on Decision and Control*, pages 1073–1080, 1984.