Updatable Strategy Logic

Christophe Chareton, Julien Brunel, David Chemouil

UPDATABLE STRATEGY LOGIC

CHRISTOPHE CHARETON, JULIEN BRUNEL, AND DAVID CHEMOUIL

Abstract. In this technical report, we present a temporal multi-agent logic called Updatable Strategy Logic (USL) which subsumes the main propositions in this area, such as ATL-ATL*, ATL_{sc} and SL. These logics allow to express the capabilities of agents to ensure the satisfaction of temporal properties. USL mainly differs from SL in two ways. Semantically, the notion of strategy composition is extended to enable an agent to update (or refine) its own strategy without revoking it. Syntactically, a new operator, called unbinder, is introduced: it allows an agent to explicitly revoke a strategy, whereas it is implicitly done according to SL semantics.

We show that USL allows to express the notion of *sustainable capability* for an agent, *i.e.*, a capability that still holds even after it has been employed. This makes USL strictly more expressive than SL. We also show that the model-checking problem for USL is decidable (but non-elementary as for SL), and that it is PSPACE-complete for its memory-less version.

1. Introduction

Multi-agent logics are receiving growing interest in contemporary research. Since the seminal work of R. Alur, Th. A. Henzinger, and O. Kupferman [AHK02], increasing efforts have been made to formalize agent interactions and strategies in semantic games.

Basically, multi-agent logics enable to formulate assertions about the ability of agents to ensure temporal properties. Thus, ATL-ATL* appears as a generalization of CTL-CTL* in which the path quantifiers **E** and **A** are replaced by *strategy quantifiers*. Strategy quantifiers (the existential $\langle\!\langle A \rangle\!\rangle$ and the universal $[\![A]\!]$) have a (coalition of) agent(s) as parameter. $\langle\!\langle A \rangle\!\rangle\varphi$ means that agents in A can act so as to ensure the satisfaction of the temporal formula φ. It is interpreted in *Concurrent Game Structures* (*CGSs*), where agents make choices influencing the execution of the system. Formula $\langle\!\langle A \rangle\!\rangle\varphi$ is true if agents in A have a strategy s.t. if they play this strategy, they force the system execution to satisfy φ, whatever the other agents do.

Since these logics allow to reason about agent interactions, the ability to express strategy composition, which comes to nesting strategy quantifiers in a formula, is a major issue. In order to illustrate this aspect, let us consider the following ATL formula involving two agents a_1 and a_2:

$$\langle\!\langle a_1 \rangle\!\rangle \Box(\varphi_1 \wedge \langle\!\langle a_2 \rangle\!\rangle \Box\varphi_2) \tag{1}$$

where $\Box\varphi$ is the classical temporal operator meaning φ is always true. In ATL-ATL* semantics, the operator $\langle\!\langle a_2 \rangle\!\rangle$ drops the strategies introduced by any earlier quantifier (here $\langle\!\langle a_1 \rangle\!\rangle$). So during the evaluation of $\Box\varphi_2$, the strategy adopted by a_1 is not taken into account. Note that here and in the remaining of this article, we use a lower case to denote a single agent and a capital letter to denote a coalition (set) of agents.

ATL_{sc} [BDCLM09, DCL11], while keeping the ATL syntax, adapts the semantics in order to interpret formulas in a context which stores strategies introduced by earlier quantifiers.

Strategy Logic (SL [MMV10, MMPV11, CHP10]) is another interesting proposition, in which the ATL operator $\langle\!\langle a \rangle\!\rangle$ is split into two different operators: an existential quantifier over strategies $\langle\!\langle x \rangle\!\rangle$, where x is a strategy variable, and a binder (a, x), which stores into a context the information that a plays along the strategy instantiating variable x (we write σ_x such a strategy in the remaining of this paper). In

order to illustrate the SL syntax, consider the following SL formula that is equivalent to formula 1 when the latter is considered as an $\mathrm{ATL}_{\mathrm{sc}}$ formula:

$$\langle\!\langle x_1 \rangle\!\rangle (a_1, x_1) [\![x_2]\!] (a_2, x_2) \dots [\![x_k]\!] (a_k, x_k) \Box (\varphi_1 \wedge \langle\!\langle x_2 \rangle\!\rangle (a_2, x_2) \Box \varphi_2) \tag{2}$$

where $[\![x]\!]$ is the universal quantifier over strategy variables (dual of $\langle\!\langle x \rangle\!\rangle$) and $\{a_1, \dots a_k\}$ is the set of agents in the system under consideration.

In SL, the composition of strategies is defined in a quite natural way (as far as the strategies concern distinct agents) through the semantics of the nesting of binders. For instance, in formula 2, when evaluating $\Box\varphi_2$, a_1 remains bound to strategy σ_{x_1} except if a_1 and a_2 are the same agent. In which case the binder (a, x_2) unbinds a from σ_{x_1} before binding her to σ_{x_2}. Indeed, when evaluating a binder (a, x), if agent a is bound to a strategy in the current context, she automatically revokes it to play along σ_x.

By automatically revoking an agent's strategy before binding her to a new one, SL prevents from reasoning about an agent refining its own strategy. In particular, it is not possible to express what we call *sustainable capability*, *i.e.* an agent's capability which remains active even when already employed by this very agent. (We also say that an agent having the sustainable capability to enforce both logical values of a given property has *sustainable control* on this property.) This will be described more precisely in the next section.

In this paper, we present Updatable Strategy Logic (USL), a logic obtained from SL by making two main evolutions. Semantically, the notion of strategy composition is extended to enable an agent to update (or refine) its own strategy without revoking it. Syntactically, in addition to the binder $(a \triangleright x)$, which keeps track of the strategies previously bound to agent a (note our notation slightly differs from SL), a new operator $(a \not\triangleright x)$, called unbinder, allows an agent to explicitly revoke a strategy.

Incidentally, USL does not have the following syntactic constraint that holds in SL: all the agents need to be explicitly bound to a strategy before evaluating a temporal formula. Because of this constraint, the agents that were not mentioned in ATL formula 1 are bound to a universally-quantified strategy in SL formula 2. In addition to the unnecessary important size of formulas, this raises the problem of modular specifications: a single informal specification can give rise to various formulas depending on the number of agents of the system that are not mentioned in the specification. In USL, thanks to a flexible treatment of contexts, we avoid this constraint. As an illustration, if agents a_1 and a_2 are not the same, the SL formula 2 would be specified as the following equivalent USL formula, which does not depend on the number of agents:

$$\langle\!\langle x_1 \rangle\!\rangle (a_1 \triangleright x_1) \Box (\varphi_1 \wedge \langle\!\langle x_2 \rangle\!\rangle (a_2 \triangleright x_2) \Box \varphi_2) \tag{3}$$

The remaining of this paper is organized as follows: Sect. 2 informally introduces USL operators and semantics. It also further explains the notions of sustainable capability and control. Sect. 3 gives the syntax and the semantics of USL, and a restricted semantics with *memory-less strategies* (we call the resulting logic USL$^{\circ}$). Sect. 4 studies the expressive power of USL relatively to SL. Sect. 5 provides the following results about model-checking: for USL it is non-elementarily decidable, and for USL$^{\circ}$ it is PSPACE-complete.

2. Updatable Strategy Logic (USL)

Let us consider the natural language property *a can always ensure p in the next state*. In SL, if a is the only agent, it is expressed by formula 4:

$$\langle\!\langle x_1 \rangle\!\rangle (a, x_1) \Box (\langle\!\langle x_2 \rangle\!\rangle (a, x_2) \mathbf{X} p) \tag{4}$$

where $\mathbf{X}$ is the temporal operator meaning *in the next state*. Let us compare formula 4 with the similar formula in USL:

$$\langle\!\langle x_1 \rangle\!\rangle (a \triangleright x_1) \Box (\langle\!\langle x_2 \rangle\!\rangle (a \triangleright x_2) \mathbf{X} p) \tag{5}$$

In formula 4, subformula $\langle\!\langle x_2 \rangle\!\rangle (a, x_2)\mathbf{X}p$ states that a can adopt a strategy σ_{x_2} that ensures $\mathbf{X}p$, even if σ_{x_2} is in contradiction with σ_{x_1} and makes her lose her ability to ensure $\mathbf{X}p$ later on. In other words, formula 4 means that a can ensure $\mathbf{X}p$ once, and decides the moment she will. On the contrary, USL requires that σ_{x_1} and σ_{x_2} can be composed. So the strategy σ_{x_2} does not revoke σ_{x_1}, according to which a has the capability to ensure $\mathbf{X}p$ later on in the execution. Thus, formula 5 expresses that a is able to ensure $\mathbf{X}p$ as many times as she wants.

The USL formula that is equivalent to formula 4 is actually formula 6, in which a is explicitly unbound from σ_{x_1} and then bound to σ_{x_2}:

$$\langle\!\langle x_1 \rangle\!\rangle (a \triangleright x_1)\Box(\langle\!\langle x_2 \rangle\!\rangle (a \not\triangleright x_1)(a \triangleright x_2)\mathbf{X}p) \tag{6}$$

This formula holds in state s_0 of model $\mathcal{M}_1$ illustrated by Fig. 1. The transitions are labelled the following way: let s, s' be two states and c a choice, then the transition from s to s' is labelled with c iff a can force it by playing choice c. Indeed, by playing strategy *always-play-*c_1, a remains in a position where she can ensure p. However, as soon as she ensures p, she loses her capability. She can achieve $\mathbf{X}p$ when she wants, but only once: her capability is not sustainable. For this reason, formula 5, which expresses the capability to ensure p as many times as agent a wants, does not hold in state s_0 of $\mathcal{M}_1$.

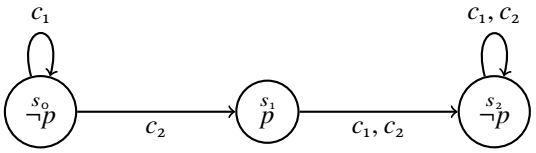

FIGURE 1. Model $\mathcal{M}_1$

Let us now consider the model $\mathcal{M}_2$ from Fig.2. Again, choices from a label the compatible transitions. Note that they are not deterministic: from a given state, a choice may not decide on the successor, even in a model with one single agent. In USL indeed, when all the agents give their choices in the current state, it does not uniquely determine the successor state.

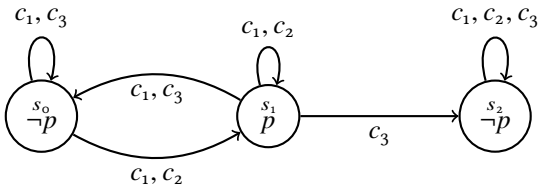

FIGURE 2. Model $\mathcal{M}_2$

In $\mathcal{M}_2$, formula 5 is true at s_0 with the strategy *always-play-*c_1 instantiating x_1. Indeed, this strategy can be refined by making the additional choice c_2 (which ensures $\mathbf{X}p$) at any time. With this strategy, a always remains capable of ensuring $\mathbf{X}p$, as many times as she wants: her capability to do so is sustainable.

A specific kind of sustainable capability arises for agent a when she is sustainably capable of deciding whether or not a property ψ holds. We speak of *sustainable control*: a sustainably controls the value of ψ. Formula 7 expresses that a has sustainable control on atomic proposition p.

$$\langle\!\langle x \rangle\!\rangle (a \triangleright x)\Box(\langle\!\langle y \rangle\!\rangle (a \triangleright y)\mathbf{X}p \wedge \langle\!\langle y \rangle\!\rangle (a \triangleright y)\mathbf{X}\neg p) \tag{7}$$

We show in Sect. 4 that the USL formula 7 cannot be satisfied in the class of SL models (because of their determinism). Furthermore, even with a natural extension of SL semantics to non-deterministic models, formula 7 is not expressible in SL.

Note that reasoning about strategies that can be revoked by later introduced strategies can be useful and fits well with the classical interpretation of branching-time operators in CTL-CTL*, where quantifiers do not restrain the possible execution but delimit those of the states and paths that are under focus. As noticed in [ÅGJ07], it corresponds to revocable strategies, where agents are not committed to strategies but are free to change them. In USL, we propose a concept of strategy composition that does not automatically revoke previous strategies in order to express sustainable capabilities, and unify it with the classical branching-time mechanisms of strategies revocation thanks to the unbinder operator: in USL, strategies are both updatable and revocable.

The following section gives the syntax and the semantics of USL.

3. Syntax and semantics

In this section we present the syntax and semantics of USL, together with the related definitions they require.

3.1. **Syntax.**

Definition 1. *Let Ag be a set of agents, At a set of propositions and X a set of variables, USL (Ag, At, X) is given by the following grammar:*

- *State formulas:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle x \rangle\!\rangle \varphi \mid (A \triangleright x)\psi \mid (A \not\triangleright x)\psi$$

- *Path formulas:*

$$\psi ::= \varphi \mid \neg\psi \mid \psi \wedge \psi \mid \psi \mathbf{U} \psi \mid \mathbf{X}\psi$$

where $p \in At$, $A \subseteq Ag$, $x \in X$. Propositional connective $\vee$ and constants $\top$ and $\bot$ are defined as usual. We also use the common temporal abbreviations $\Diamond$ and $\Box$, defined as follows: $\Diamond\varphi \triangleq \top\mathbf{U}\varphi$ and $\Box\varphi \triangleq \neg\Diamond\neg\varphi$.

A formula where each bound variable is previously quantified is called a *sentence*. We define the set of *subformulas* of a formula φ as usual and we call its *length* $|\varphi|$ the number of its subformulas. We define inductively a notion of *free variables* $Fv(\varphi)$ as follows:

- $Fv(p) = \emptyset$, for $p \in At$
- $Fv(\neg\varphi) = Fv(\mathbf{X}\varphi) = Fv((A \not\triangleright x)\varphi) = Fv(\varphi)$
- $Fv(\varphi_1 \wedge \varphi_2) = Fv(\varphi_1 \mathbf{U} \varphi_2) = Fv(\varphi_1) \cup Fv(\varphi_2)$
- $Fv(\langle\!\langle x \rangle\!\rangle \varphi) = Fv(\varphi) - \{x\}$
- $Fv((A \triangleright x)\varphi) = Fv(\varphi) \cup \{x\}$

A *sentence* is a state formula φ s.t. $Fv(\varphi) = \emptyset$.

3.2. **Semantics.**

In this section we define the models of USL, which extends classical models of ATL and SL with non-determinism. We also define the related semantic notions.

Definition 2. *A Non-deterministic Alternating Transition System (NATS) is a tuple $\mathcal{M} = \langle Ag, M, At, v, Ch\rangle$ where:*

- *M is a set of states, called the domain of the NATS, At is the set of atomic propositions and v is a valuation function, from M to $\mathcal{P}(At)$.*
- *Ch: $Ag \times M \to \mathcal{P}(\mathcal{P}(M))$ is a choice function mapping a pair $\langle agent, state\rangle$ to a non-empty family of choices of possible next states. It is such for every state $s \in M$ and for every agent a_1 and a_2 in Ag, for every $c_1 \in Ch(a_1, s)$ and $c_2 \in Ch(a_2, s)$, $c_1 \cap c_2 \neq \emptyset$.*

We call a finite sequence of states in M a *track* τ. The last element of a track τ is denoted by $last(\tau)$. The set of tracks that are possible in $\mathcal{M}$ is denoted by $track_{\mathcal{M}} : \tau = s_0 s_1 \dots s_k \in track_{\mathcal{M}}$ iff for every $i < k$, for every $a \in Ag$, there is $c_a \in \mathcal{P}(M)$ s.t. $c_a \in Ch(a, s_i)$ and $s_{i+1} \in c_a$. Similarly, an infinite sequence of states that are possible in $\mathcal{M}$ (all its prefixes are in $track_{\mathcal{M}}$) is called a *path* (in $\mathcal{M}$). Let λ

be a path, then for every $i \in \mathbb{N}$, we write λ_i its i^{th} element. We also write λ^k the path s.t. for every $i \in \mathbb{N}$, $\lambda^k_i = \lambda_{k+i}$.

A *strategy* is a function σ from $Ag \times track_{\mathcal{M}}$ to $\mathcal{P}(M)$ s.t. for every $(a, \tau) \in Ag \times track_{\mathcal{M}}$, $\sigma(a, \tau) \in Ch(a, last(\tau))$. We note *Strat* the set of strategies in a model. Now, *out* (s, σ) denotes the set of outcomes of σ from s, *i.e.* the set of paths in which the agents have been using σ: let σ be a strategy and $\lambda = \lambda_o\lambda_1 \dots$ an infinite sequence over M, then $\lambda \in out(s, \sigma)$ iff λ is a path in $\mathcal{M}$, $s = \lambda_o$ and for every $n \in \mathbb{N}$, $\lambda_{n+1} \in \bigcap_{a \in Ag} \sigma(a, \lambda_o \dots \lambda_n)$.

A *context* κ is a finite sequence upon $(\mathcal{P}(Ag) \times X)$, representing the active bindings. Let κ_1 and κ_2 be two different contexts, then we note $\kappa_1 \cdot \kappa_2$ their concatenation. A *memory* μ is a partial function from X to *Strat*, mapping quantified strategy variables to the corresponding strategy instantiation. A *plan* π is a pair of a memory and a context. Just as a strategy does, a plan defines a function from $track_{\mathcal{M}}$ to $\mathcal{P}(M)$. We use the same notation for the plan itself and its induced function. Let $\kappa_\emptyset$ be the empty sequence upon $(\mathcal{P}(Ag) \times X)$, then $(\mu, \kappa_\emptyset)(\tau) = M$, $(\mu, (A, x))(\tau) = \bigcap_{a \in A} \mu(x)(a, \tau)$ if $A \neq \emptyset$, else M, and $(\mu, \kappa \cdot (A, x))(\tau) = (\mu, \kappa)(\tau) \cap (\mu, (A, x))(\tau)$ if this intersection is not empty. Otherwise (which means the context stores contradictory strategies for the same agent) $(\mu, \kappa \cdot (A, x))(\tau) = (\mu, \kappa)(\tau)$. Now we can define the outcomes of a plan π, *out* (π), as the set of paths derived from π: let $\lambda = \lambda_o, \lambda_1, \dots$ be an infinite sequence over M, then $\lambda \in out(s, \pi)$ iff λ is a path in $\mathcal{M}$, $s = \lambda_o$ and for every $n \in \mathbb{N}$, $\lambda_{n+1} \in \pi(\lambda_o \dots \lambda_n)$. Notice that the outcomes of a strategy can be defined as the outcomes of a plan that only represents this strategy: for every strategy σ and state s, $out(s, \sigma) = out(s, ((x \mapsto \sigma), (Ag, x)))$.

Definition 3 (Strategy and memory translation). *Let σ be a strategy and τ be a track. Then σ^τ is the strategy s.t. for every $\tau' \in track_{\mathcal{M}}$, $\sigma^\tau(\tau') = \sigma(\tau\tau')$. The notion is extended to a memory: for every μ, μ^τ is the memory with domain equal to that of μ and s.t. for every $x \in dom(\mu)$, $\mu^\tau(x) = (\mu(x))^\tau$*

We also define the following transformations of contexts and memories. Given a context κ, coalitions A and B, a strategy variable x, a memory μ and a strategy σ:

- $\kappa[A \to x] = \kappa \cdot (A \rhd x)$
- $((B, x) \cdot \kappa)[A \twoheadrightarrow x] = (B \backslash A, x) \cdot (\kappa[A \twoheadrightarrow x])$ and $\kappa_\emptyset[A \twoheadrightarrow x] = \kappa_\emptyset$
- $\mu[x \to \sigma]$ is the memory with domain $dom(\mu) \cup \{x\}$ s.t. $\forall y \in dom(\mu) \backslash \{x\}$, $\mu[x \to \sigma](y) = \mu(y)$ and $\mu[x \to \sigma](x) = \sigma$

Definition 4 (Satisfaction relation). *Let $\mathcal{M}$ be a NATS, then for every memory μ, context κ, state s and path λ:*

- *State formulas:*
  - $\mathcal{M}, \mu, \kappa, s \models p$ *iff* $p \in v(s)$, *with* $p \in At$
  - $\mathcal{M}, \mu, \kappa, s \models \neg\varphi$ *iff it is not true that* $\mathcal{M}, \mu, \kappa, s \models \varphi$
  - $\mathcal{M}, \mu, \kappa, s \models \varphi_1 \wedge \varphi_2$ *iff* $\mathcal{M}, \mu, \kappa, s \models \varphi_1$ *and* $\mathcal{M}, \mu, \kappa, s \models \varphi_2$
  - $\mathcal{M}, \mu, \kappa, s \models \langle\langle x \rangle\rangle\varphi$ *iff there is a strategy* $\sigma \in Strat$ *s.t.* $\mathcal{M}, \mu[x \to \sigma], \kappa, s \models \varphi$
  - $\mathcal{M}, \mu, \kappa, s \models (A \rhd x)\varphi$ *iff for every* λ *in* $out(\mu, \kappa[A \to x])$, $\mathcal{M}, \mu, \kappa[A \to x], \lambda \models \varphi$
  - $\mathcal{M}, \mu, \kappa, s \models (A \not\rhd x)\varphi$ *iff for all* λ *in* $out(\mu, \kappa[A \twoheadrightarrow x])$, $\mathcal{M}, \mu, \kappa[A \twoheadrightarrow x], \lambda \models \varphi$
- *Path formulas :*
  - $\mathcal{M}, \mu, \kappa, \lambda \models \varphi$ *iff* $\mathcal{M}, \mu, \kappa, \lambda_o \models \varphi$, *for every state formula* φ
  - $\mathcal{M}, \mu, \kappa, \lambda \models \neg\psi$ *iff it is not true that* $\mathcal{M}, \mu, \kappa, \lambda \models \psi$
  - $\mathcal{M}, \mu, \kappa, \lambda \models \psi_1 \wedge \psi_2$ *iff* $\mathcal{M}, \mu, \kappa, \lambda \models \psi_1$ *and* $\mathcal{M}, \mu, \kappa, \lambda \models \psi_2$
  - $\mathcal{M}, \mu, \kappa, \lambda \models \mathbf{X}\psi$ *iff* $\mathcal{M}, \mu^{\lambda_o}, \kappa, \lambda^1 \models \psi$.
  - $\mathcal{M}, \mu, \kappa, \lambda \models \psi_1 \mathbf{U} \psi_2$ *iff there is* $i \in \mathbb{N}$ *s.t.* $\mathcal{M}, \mu^{\lambda_o \dots \lambda_{i-1}}, \kappa, \lambda^i \models \psi_2$ *and for every* $o \leq j < i$, $\mathcal{M}, \mu^{\lambda_o \dots \lambda_{j-1}}, \kappa, \lambda^j \models \psi_1$

Let $\mu_\emptyset$ be the unique strategy memory with empty domain, $\kappa_\emptyset$ the empty word upon $(Ag \times X)$ and φ be a sentence in USL. Then $\mathcal{M}, s \models \varphi$ iff $\mathcal{M}, \mu_\emptyset, \kappa_\emptyset, s \models \varphi$.

Let us give the following comment over these definitions: for every plan $\pi = (\mu, \kappa)$, the definition of *out* (π) ensures that the different binders encoded in π compose their choices together, *as far as possible*. In case two contradictory choices from an agent are encoded in the plan, the priority is given to the first binding that was introduced in this plan (the left most binding in the formula). This guarantees that a formula requiring the composition of two contradictory strategies is false. For example, suppose that $\langle\langle x_1 \rangle\rangle (a \triangleright x_1)\varphi_1$ and $\langle\langle x_2 \rangle\rangle (a \triangleright x_2)\varphi_2$ are both true in a state of a model, and suppose that σ_{x_1} and σ_{x_2} necessarily rely on contradictory choices of a (this means that a cannot play in a way that ensures both φ_1 and φ_2). Then, $\langle\langle x_1 \rangle\rangle (a \triangleright x_1)(\varphi_1 \wedge \langle\langle x_2 \rangle\rangle (a \triangleright x_2)\varphi_2)$ is false in the same state of the same model. If the priority was given to the most recent binding (right most binding in the formula), the strategy σ_{x_1} would be revoked and the formula would be satisfied.

3.3. **USL^o.** USL^o is the logic obtained by modifying the semantics of USL so that the strategies only depend on the current state, rather than on a whole track. We call such strategies *memory-less strategies*. Many cases of programming and interaction situations can actually be modeled by only using memory-less strategies.

Definition 5. *A memory-less strategy is a function σ from $Ag \times M$ to $\mathcal{P}(M)$. It is s.t. for every $(a, s) \in Ag \times M, \sigma(a, s) \in Ch(a, s)$. We note $Strat^o$ the set of memory-less strategies in a model.*

All the semantic definitions for USL adapt to USL^o by simply modifying the clause for $\langle\langle x \rangle\rangle$: the strategy variable ranges over $Strat^o$ instead of *Strat* in the case of USL^o.

4. USL and SL

In this section, we compare the expressive power of SL and USL. This comparison is not straightforward because the models of both logics differ slightly: in particular SL models, called *CGSs*, are deterministic in the sense that given the current state and a choice for every agent, there is only one possible successor state, which is not the case in USL models (*NATS*).

The main results of this section are stated as follows:

- In order to show the non-strict inclusion of SL into USL, we provide an embedding of SL models (Concurrent Games Structures, or *CGSs*) and formulas into USL models and formulas. Our embedding preserves the satisfaction relation.
- In the case of one single agent (USL^1 and SL^1) the embedding is trivial. In this case, we show that:
  - Sustainable control is not expressible under deterministic models, neither in USL nor in SL. Then one needs the non-determinism of *NATSs* to express it.
  - Sustainable control expressed by formula 7 is not expressible under a natural extension of SL semantics to *NATSs*.

4.1. **Embedding of SL into USL.** The embedding of SL into USL consists in a parallel transformation from SL models and formulas to USL models and formulas. This transformation preserves the satisfaction relation. Let us first bring the necessary definitions related to SL:

4.1.1. *SL.* This paragraph only gives a brief presentation of SL syntax and semantics. The non familiar reader may refer to [MMPV11] for a complete presentation of this framework.

Definition 6 (SL syntax)**.** *Given a set Ag of agents, a set At of atomic propositions and a set X of strategy variables, the syntax of SL is defined by the following grammar:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U} \varphi \mid \varphi \mathbf{R} \varphi \mid \langle\langle x \rangle\rangle \varphi \mid [\![x]\!] \varphi \mid (a, x)\varphi$$

where a is an agent and x ranges over a finite set X of strategy variables. The sentences of SL are the formulas with no free variable and s.t. every temporal subformula (of type $\mathbf{X}\psi_1, \psi_1 \mathbf{U} \psi_2$ or $\psi_1 \mathbf{R} \psi_2$) is in the scope of at least one binder (a, x) for each agent a.

Here is the definition of *CGSs*:

Definition 7. *A CGS is a tuple $\mathcal{G} = \langle Ag, M, At, v, Ac, Tr\rangle$ where :*

- *Ag, M, At and v are as in CGSs.*
- *Ac is a set of actions.*
- *Let $DC = Ac^{Ag}$ be the set of decisions,* i.e. *the set of total functions from agents to actions. Then $Tr : M \times DC \to M$ is the transition function.*

Note that, modulo the interpretation of actions as the set of successors compatible with each agent playing them (for every $a \in Ag$, for every ac $\in Ac$, the choice corresponding to (a, ac) is $\{Tr(Dc) \mid Dc \in Ac^{Ag}, Dc(a) = ac\}$), *CGSs* are a particular case of *NATSs* (the case where a vector of one choice per agent from a given state uniquely determines a successor).

Definition 8. *Let $\mathcal{G}$ be a CGS, the set $track_{\mathcal{G}}$ is the set of paths $\lambda_0\lambda_1\dots$ s.t. for every $n \in \mathbb{N}$ there is $Dc \in DC$ s.t. $Tr(\lambda_n, Dc) = \lambda_{n+1}$. A strategy in CGS $\mathcal{G}$ is a partial function from $track_{\mathcal{G}}$ to the set of actions. We note Strat the set of strategies in a model:*

$$Strat = \{\sigma : track_{\mathcal{G}} \to Ac\}$$

We also call s-total a strategy that is defined upon the whole set of sequences beginning with s. A strategy context is a partial function from $Ag \cup X$ to Strat. It is s-total if it defines only s-total strategies and it is complete if Ag is included in its domain.

Note that strategies for SL have tracks as single parameter, so that a strategy binds each agent of a coalition to the same actions. Then the set of possible strategies for a coalition A does not match the product of the sets of possible strategies for agents in A. This is the main reason why the embedding from SL to USL is not trivial.

The semantics of SL is given as a relation between a formula and a triple $(\mathcal{G}, \kappa, s)$ where:

- $\mathcal{G}$ is a *CGS*.
- κ is a strategy context over $\mathcal{G}$.
- s is a state in M, the domain of $\mathcal{G}$.

The transition function *Tr* being defined over the set of decisions, a strategy context must be complete and s-total to determine a transition from a given state s. Under this condition we can write $(\kappa, s)^n$ the pair defined by:

- $(\kappa, s)^0 = (\kappa, s)$
- for every $i \in \mathbb{N}$, $(\kappa, s)^{i+1} = (\kappa_{i+1}, s_{i+1}) = (\kappa_i^{s_i}, Tr(s_{i+1}, \kappa_i(Ag)))$

where the notation κ^τ, for κ a strategy context and τ a track, is as in Def. 3. Again, the definition for the semantics of SL proceeds in two steps:

Definition 9 (Satisfaction relation for SL)**.** *Let $\mathcal{G}$ be a CGS, κ a strategy context for $\mathcal{G}$ and s a state in it. Then:*

(1) $\mathcal{G}, \kappa, s \models_{SL} p$ iff $p \in \lambda(s)$, with $p \in At$.
(2) For every formula φ, φ_1 and φ_2 :
  (a) $\mathcal{G}, \kappa, s \models_{SL} \neg\varphi$ iff it is not true that $\mathcal{G}, \kappa, s \models_{SL} \varphi$.
  (b) $\mathcal{G}, \kappa, s \models_{SL} \varphi_1 \wedge \varphi_2$ iff $\mathcal{G}, \kappa, s \models_{SL} \varphi_1$ and $\mathcal{G}, \kappa, s \models_{SL} \varphi_2$.
  (c) $\mathcal{G}, \kappa, s \models_{SL} \varphi_1 \vee \varphi_2$ iff $\mathcal{G}, \kappa, s \models_{SL} \varphi_1$ or $\mathcal{G}, \kappa, s \models_{SL} \varphi_2$.
(3) For every variable x and formula φ :
  (a) $\mathcal{G}, \kappa, s \models_{SL} \langle\langle x\rangle\rangle\varphi$ iff there is an s-total strategy f s.t. $\mathcal{G}, \kappa[x \to f], s \models_{SL} \varphi$.
  (b) $\mathcal{G}, \kappa, s \models_{SL} [\![x]\!]\varphi$ iff for every s-total strategy f, $\mathcal{G}, \kappa[x \to f], s \models_{SL} \varphi$.
(4) For every agent a, variable x and formula φ, $\mathcal{G}, \kappa, s \models_{SL} (a, x)\varphi$ iff $\mathcal{G}, \kappa[\kappa(x)\backslash\kappa(a)], s \models_{SL} \varphi$.
(5) If κ is a complete strategy context, then for every formula φ, φ_1 and φ_2 :
  (a) $\mathcal{G}, \kappa, s \models_{SL} \mathbf{X}\varphi$ iff $\mathcal{G}, (\kappa, s)^1 \models_{SL} \varphi$.

(b) $\mathcal{G}, \kappa, s \models_{SL} \varphi_1 \mathbf{U} \varphi_2$ *iff there is* $i \in \mathbb{N}$ *s.t.* $\mathcal{G}, (\kappa, s)^i \models_{SL} \varphi_2$ *and, for every index* $j \in \mathbb{N}$ *s.t.* $0 \leq i < j$, $\mathcal{G}, (\kappa, s)^j \models_{SL} \varphi_1$.

(c) $\mathcal{G}, \kappa, s \models_{SL} \varphi_1 \mathbf{R} \varphi_2$ *iff for every index* $i \in \mathbb{N}$, *it holds that* $\mathcal{G}, (\kappa, s)^i \models_{SL} \varphi_2$ *or there is an index* $j \in \mathbb{N}$ *s.t.* $0 \leq j \leq i$, $\mathcal{G}, (\kappa, s)^j \models_{SL} \varphi_1$

Where $\kappa[\kappa(x) \backslash \kappa(a)]$ *is obtained from* κ *by giving for* a *the value* $\kappa(x)$.

Definition 10 (Satisfaction of a sentence in a state of a model)**.**

Let $\mathcal{G}$ *be a CGS and* s *a state in its domain. Let also* φ *be a formula in SL.* φ *is true in* $\mathcal{G}$ *at* s, *we note* $\mathcal{G}, s \models_{SL} \varphi$, *if and only if* $\mathcal{G}, \kappa_\emptyset, s \models_{SL} \varphi$, *where* $\kappa_\emptyset$ *is the context with empty domain.*

The required definitions being given, we can assert the existence of our embedding:

Theorem 1. *There is a transformation which maps each CGS* $\mathcal{G}$ *to a NATS* $\mathcal{M}_\mathcal{G}$ *and each formula* φ *in SL to a formula* φ_{USL} *in USL s.t. for every CGS* $\mathcal{G}$ *and for every* $\varphi \in SL$, $\mathcal{G} \models \varphi$ *iff* $\mathcal{M}_\mathcal{G} \models \varphi_{USL}$. *Furthermore, if we consider a single agent in the languages, the transformation of models reduces to the interpretation of actions as choices.*

The proof of theorem 1 is detailed in in the following paragraphs. The differences between SL and USL lie both in the definition of strategies in SL semantics and in the differences of interpretation for the binding operator. We treat successively these two points: first we define an internal transformation for SL, by which the constraints of agents playing the same choices are expressed in the syntax. Then we define a new operator in USL that is equivalent to SL binding, and we show the equivalence.

4.1.2. *About the constraints of uniform strategy playing.* First, we define both the syntactical and semantical transformations that enable to express, by help of new atomic symbols, the constraints of agents playing uniform strategies.

- Syntax: transform φ to φ'. The transformation consists in identifying, in the syntax of the formula, those of the coalitions that have to play along the same strategy. It enables to represent a part of the information hold by the current context in every subformula.
  - First erase all $\vee$, $[\![x]\!]$ and $\mathbf{R}$ operators from φ, by use of the equivalences $\psi_1 \vee \psi_2$ iff $\neg(\neg\psi_1 \wedge \neg\psi_2)$, $[\![x]\!]\psi$ iff $\neg\langle\!\langle x \rangle\!\rangle \neg\psi$ and $\psi_1 \mathbf{R} \psi_2$ iff $\neg(\neg\psi_1 \mathbf{U} \neg\psi_2)$.
  - Let $\equiv$ be the equivalence relation over $Ac^{Ag} = DC$ given by: for every $\kappa, \kappa' \in DC$, $\kappa \equiv \kappa'$ iff $\forall a, b \in Ag, (\kappa(a) = \kappa(b))$ iff $(\kappa'(a) = \kappa'(b))$. Let also $[DC]_\equiv$ be the partition of DC over $\equiv$. Then for every $P \in [DC]_\equiv$, add a new proposition $\overline{P}$ in the language.
  - For each subformula ψ of φ:
    - If $\psi := \mathbf{X}\psi_1$ change it for $\mathbf{X}(\psi_1 \wedge \overline{P_{\psi_1}})$ where P_{ψ_1} is given by the set of binders ψ is in the scope of (because φ is a sentence they completely define P_{ψ_1}).
    - If $\psi := \psi_1 \mathbf{U} \psi_2$ change it for $\psi_2 \vee \mathbf{X}(\psi_1 \wedge \overline{P_{\psi_2}}) \mathbf{U} (\psi_2 \wedge \overline{P_{\psi_2}})$
    - Else do not change ψ.
- Semantics: change $\mathcal{G}$ to $\mathcal{G}'$. Intuitively, we make a different copy of the domain M for each $p \in DC$. Then for each agent a, the choices of a in the new *CGS* are the set of outcomes of the function Tr times the set of copies of M where a plays ac, for each $ac \in Ac$. Formally:

  Let $\mathcal{G} = \langle Ag, M, At, v, Ac, Ch \rangle$. Then $\mathcal{G}'$ is the *CGS* $\langle Ag, M', At', v', Ac, Tr' \rangle$ where:
  - $M' = M \times DC$
  - $At' = At \cup \{\overline{P} \mid P \in [DC]_\equiv\}$
  - for every $(s, Dc) \in M'$, $v'(s, Dc) = v(s) \cup \{\overline{[Dc]_\equiv}\}$, where $[Dc]_\equiv$ is the element of $[DC]_\equiv$ induced by Dc.
  - for every $(s, Dc) \in M'$, for every $Dc' \in Ac^{Ag}$, $Tr'((s, Dc), Dc') = (Tr(s, Dc'), Dc')$.

Then we have the following lemma:

Lemma 1.

$$\mathcal{G} \models_{SL} \varphi \text{ iff } \mathcal{G}' \models_{SL} \varphi'$$

Proof. One simply replaces every subformula of type $\psi_1 \mathbf{U} \psi_2$ in φ by $\psi_2 \vee \mathbf{X}(\psi_1 \mathbf{U} \psi_2)$. Then the equivalence is obtained by induction over φ complexity. □

The next intermediary step is to consider $\mathcal{G}'$ as a *NATS* and use a semantics for SL in *NATS*. Viewing $\mathcal{G}'$ as a *NATS* only consists in interpreting each 3-uple (a, s, ac) of an agent, a state and an action as the set of potential successor states when a performs ac from s. The semantics for SL in *NATSs* holds the following case for (a, x): $\mathcal{G}', \mu, \kappa, s \models_{NATS} (a, x)\psi$ iff $\mathcal{G}', \mu, \kappa[\kappa(x) \backslash \kappa(a)], s \models (a, x)\psi$. Since we are using it only in the model $\mathcal{G}'$ with deterministic choices, $\models_{NATS}$ does not need further definition so far. An extension of $\models_{NATS}$ for every *NATSs* for languages with one agent is given in sub-section 4.3. Both semantics are obviously equivalent for the evaluation of φ in $\mathcal{G}'$. Now, we can come to the translation of SL binder. To achieve this, we define a new operator in USL.

4.1.3. *Translation into USL.* Let $X' \subseteq X$ be a set of variables, and let us abbreviate by $(A \not\triangleright X')$ the sequence of operators $(A \not\triangleright x_1), \ldots, (A \not\triangleright x_n)$ where $X' = \{x_1, \ldots x_n\}$, for any coalition A (note that the semantics of the sequence $(A \not\triangleright x_1), \ldots, (A \not\triangleright x_n)$ is invariant upon the ordering of the x_is). We define the new operator $[A \triangleright x]$ by : for any formula φ, coalition A and variable x, $[A \triangleright x]\varphi \triangleq (A \not\triangleright X)(A \triangleright x)\varphi$. Then $[A \triangleright x]$ is the counterpart for the binder in SL: A is unbound from all its current strategies and then bound to the sole strategy σ_x. (Note that by use of a macro for $(Ag \not\triangleright X)$ $(A \triangleright x)$, an ATL-like capability operator is also definable for USL).

Now we replace inductively, from innermost to outermost subformulas, all subformulas $(a \triangleright x)\psi$ of φ' by $[a \triangleright x]\, \psi$, and we call φ_{USL} the resulting formula. We have:

Lemma 2.

$$\mathcal{G}' \models_{NATS} \varphi' \text{ iff } \mathcal{G}' \models \varphi_{USL}$$

Proof. Straightforward, since $[a \triangleright x]$ in USL is interpreted the same way as (a,x) in SL. □

The two preceding lemmas prove theorem 1.

4.2. **Sustainable control is not expressible over deterministic models.** Let us recall the embedding used for proof of theorem 1 applied from SL with one agent (SL^1) to USL with one agent (USL^1):

- For the syntactical part, formula φ is turned into formula φ_{USL} obtained by, for every subformula of φ of type $(a, x)\psi$:
  - If $(a, x)\psi$ is not itself a subformula of another subformula of this type in φ, then doing nothing.
  - Otherwise, replacing $(a, x)\psi$ by $(a \not\triangleright x_1) \ldots (a \not\triangleright x_k)(a \triangleright x)\psi'$, where $\{x_1, \ldots x_k\}$ is the set of variables in the language.
- For the semantical part: let $\mathcal{G} = \langle\{a\}, M, At, v, Ac, Tr\rangle$, we use it as the *NATS* $\mathcal{M}_{\mathcal{G}} = \langle\{a\}, M, At, v, Ch\rangle$ s.t. for every $s \in M$, $Ch(a, s) = \{\{Tr(s, \langle(a, ac)\rangle)\}\}_{ac \in Ac}$.

Since for SL^1 the transformation of models from SL to USL reduces to the interpretation of actions as choices, elements of comparisons can be given between the expressive powers of SL^1 and USL^1.

In the following paragraphs we show that deterministic models cannot satisfy a formula of sustainable control either in SL^1 or in USL^1. So, the non-determinism of *NATSs* is necessary. We also show that the extension of SL^1 semantics to non deterministic models does not enable to express sustainable control: such properties also need the composition of choices introduced in USL^1.

Let us designate by $DNATS^1$ a *NATS* with one agent and with deterministic choices: a *NATS* $\mathcal{M} = \langle\{a\}, M, At, v, Ch\rangle$ is a $DNATS^1$ iff for every $s \in M$, for every $c \in Ch(a, s)$, c is a singleton.

Modulo the interpretation of actions as choices, *CGSs* for one agent are $DNATS^1$s, s.t. we can directly interpret SL^1 in $DNATS^1$s.

Let us now consider the formula expressing sustainable control of a over property p:

$$\theta_\infty := \langle\!\langle x \rangle\!\rangle (a \triangleright x) \Box (\langle\!\langle y \rangle\!\rangle (a \triangleright y) \mathbf{X} p \wedge \langle\!\langle y \rangle\!\rangle (a \triangleright y) \mathbf{X} \neg p)$$

We have the following lemma:

Lemma 3. *for every DNATS*^{1} $\mathcal{M}$*, for every* $s \in M$ $\quad \mathcal{M}, s \not\models \theta_\infty$

Proof. Suppose that there is a *DNATS*^{1} $\mathcal{M}$ and a state s s.t. $\mathcal{M}, s \models \theta_\infty$. Then there is a strategy σ_x s.t. $\mathcal{M}, \langle (x \to \sigma_x)\rangle, (a,x), s \models \Box(\langle\!\langle y\rangle\!\rangle(a \triangleright y)\mathbf{X}p \wedge \langle\!\langle y\rangle\!\rangle(a \triangleright y)\mathbf{X}\neg p)$. Because $\mathcal{M}$ is a *DNATS*^{1}, a playing σ_x determines an unique path $out(s, \langle (x \to \sigma_x)\rangle, (a,x)) = \lambda = \lambda_0\lambda_1 \ldots$ in $\mathcal{M}$.

Now, next steps in the evaluation of $\mathcal{M}, s \models \theta_\infty$ require, for every $i \in \mathbb{N}$, that $\mathcal{M}, \langle (x \to \sigma_x)\rangle, (a,x), \lambda_i \models \langle\!\langle y\rangle\!\rangle(a \triangleright y)\mathbf{X}p$ and $\mathcal{M}, \langle (x \to \sigma_x)\rangle, (a,x), \lambda_i \models \langle\!\langle y\rangle\!\rangle(a \triangleright y)\mathbf{X}\neg p$. And the following steps require for every $i \in \mathbb{N}$, that there are strategies σ_{y_1} and σ_{y_2} s.t. $\mathcal{M}, \langle (x \to \sigma_x)(y \to \sigma_{y_1})\rangle, (a,x)\cdot(a,y), \lambda_{i+1} \models p$ and $\mathcal{M}, \langle (x \to \sigma_x)(y \to \sigma_{y_2})\rangle, (a,x)\cdot(a,y), \lambda_{i+1} \models \neg p$.

But, for every y, $out(\lambda_i, \langle (x \to \sigma_x), (y \to \sigma_{y_k})\rangle, (a,x)\cdot(a,y_k))$ is a non-empty set of paths included in $\{\lambda^{i+1}\}$, so it is equal to $\{\lambda^{i+1}\}$ itself. Eventually, the satisfaction requires that, for every $i \in \mathbb{N}$, $p \in \nu(\lambda_{i+1})$ and $p \notin \nu(\lambda_{i+1})$, which gives a contradiction. □

We deduce from the previous lemma that sustainable control is not expressible by SL^1 formalism: suppose it is expressible by $\Theta \in$ SL^1, then there is a *CGS* $\mathcal{G}$ with state s s.t. $\mathcal{G}, s \models_{\text{SL}} \Theta$. Then $\mathcal{M}_\mathcal{G}, s \models \Theta_{\text{USL}}$, so $\mathcal{M}_\mathcal{G}, s \models \theta_\infty$, in contradiction with lemma 3.

4.3. **Sustainable control is not expressible in SL^1 over *NATSs*.** From preceding paragraphs we have that sustainable control is not expressible under deterministic models. Here we show that the non-determinacy of *NATSs* is not sufficient to express it. One also needs the compositional interpretation of binders as defined for USL semantics.

The following argument uses an extension of SL^1 semantics to *NATSs*. We already have a straightforward interpretation of SL^1 over *DNATS*^{1}*s*. The generalization to *NATSs* needs to define an interpretation of the specifications of non determined transitions. Indeed, the difference between *DNATS*^{1}*s* and *NATSs* lies in the fact that under *DNATS*^{1}*s*, a plan π defining a strategy for a induces an unique possible path λ, so that evaluating temporal formula φ in *DNATS*^{1}*s* follows the classical definition for semantics of LTL under paths. Under *NATSs*, π induces a non-empty set of paths with possibly more than one element. We generalize the semantics by following the semantics of LTL under Kripke models, taking the universal quantification over paths in $out(s,\pi)$. We get the following definition:

Definition 11 (Semantics for SL^1 under *NATSs*)**.** *Let* $\mathcal{M}$ *be a NATS for a language with one agent,* s *a state in* $\mathcal{M}$*,* (μ,κ) *a plan for* $\mathcal{M}$ *and* φ, φ_1 *and* φ_2 *formulas in SL*^{1} *language, then:*

- $\mathcal{M}, \mu, \kappa, s \models_{NATS} p$ *iff* $p \in \nu(s)$*, with* $p \in At$*.*
- $\mathcal{M}, \mu, \kappa, s \models_{NATS} \neg\varphi$ *iff it is not true that* $\mathcal{M}, \mu, \kappa, s \models_{NATS} \varphi$*.*
- $\mathcal{M}, \mu, \kappa, s \models_{NATS} \varphi_1 \wedge \varphi_2$ *iff* $\mathcal{M}, \mu, \kappa, s \models_{NATS} \varphi_1$ *and* $\mathcal{M}, \mu, \kappa, s \models_{NATS} \varphi_2$*.*
- $\mathcal{M}, \mu, \kappa, s \models_{NATS} \varphi_1 \vee \varphi_2$ *iff* $\mathcal{M}, \mu, \kappa, s \models_{NATS} \varphi_1$ *or* $\mathcal{M}, \mu, \kappa, s \models_{NATS} \varphi_2$*.*
- $\mathcal{M}, \mu, \kappa, s \models_{NATS} \langle\!\langle x\rangle\!\rangle\varphi$ *iff there is a strategy* $\sigma \in Strat$ *s.t.* $\mathcal{M}, \mu[x \to \sigma], \kappa, s \models_{NATS} \varphi$*.*
- $\mathcal{M}, \mu, \kappa, s \models_{NATS} [\![x]\!]\varphi$ *iff for every strategy* $\sigma \in Strat$*,* $\mathcal{M}, \mu[x \to \sigma], \kappa, s \models_{NATS} \varphi$*.*
- $\mathcal{M}, \mu, \kappa, s \models_{NATS} (a,x)\varphi$ *iff* $\mathcal{M}, \mu, \kappa[x\backslash\kappa(a)], s \models_{NATS} \varphi$*.*
- $\mathcal{M}, \mu, \kappa, s \models_{NATS} \mathbf{X}\varphi$ *iff for every* $\lambda \in out(s,(\mu,\kappa))$*,* $\mathcal{M}, \mu^{\lambda_0}, \kappa, \lambda^1 \models_{NATS} \varphi$*.*
- $\mathcal{M}, \mu, \kappa, \lambda \models_{NATS} \varphi_1\mathbf{U}\varphi_2$ *iff for every* $\lambda \in out(s,(\mu,\kappa))$*, there is* $i \in \mathbb{N}$ *s.t.* $\mathcal{M}, \mu^{\lambda_0\ldots\lambda_{i-1}}, \kappa, \lambda^i \models_{NATS} \varphi_2$ *and for all* $0 \le j \le i$*,* $\mathcal{M}, \mu^{\lambda_0\ldots\lambda_{i-1}}, \kappa, \lambda^j \models_{NATS} \varphi_1$*.*
- $\mathcal{M}, \mu, \kappa, \lambda \models_{NATS} \varphi_1\mathbf{R}\varphi_2$ *iff for every* $\lambda \in out(s,(\mu,\kappa))$*,* $\mathcal{M}, \mu^{\lambda_0\ldots\lambda_{i-1}}, \kappa, \lambda^i \models_{NATS} \varphi_2$ *or there is* $j \le i$ *s.t.* $\mathcal{M}, \mu^{\lambda_0\ldots\lambda_{i-1}}, \kappa, \lambda^i \models_{NATS} \varphi_2$*.*
- *If the principal operator in* φ *is not in* $\{\mathbf{X}, \mathbf{U}, \mathbf{R}\}$*,* $\mathcal{M}, \mu, \kappa, \lambda \models_{NATS} \varphi$ *iff* $\mathcal{M}, \mu, \kappa, \lambda_0 \models_{NATS} \varphi$*.*

where $\kappa[x\backslash\kappa(a)]$ *designates the context obtained from* κ *by replacing every* (a,y) *in it by* (a,x)*.*

Now we can give the following lemma:

Lemma 4. θ_∞ *is not expressible in SL*^{1} *over NATSs.*

To prove that lemma, we proceed the following way: first, we show that if there is a formula $\Theta \in$ SL, equivalent to θ_∞ under $\models_{NATS}$, then it can be written with help of the operator $\Box$, but without $[\![x]\!]$ nor $\mathbf{U}$ nor $\mathbf{R}$. Then we show, under convenient restrictions, that the related fragment of SL satisfiability reduces to the satisfiability for first order logic. Then, a compactness argument can be applied and shows that Θ cannot be a formula in SL.

We first consider the set of formulas $\{\theta_i\}_{i\in\mathbb{N}}$, each one asserting that a is indefinitely capable to decide whether p holds or not in next state and can use this ability at least $i+1$ times. We define θ_i by induction over i:

- $\theta_0 := \langle\!\langle x\rangle\!\rangle(a,x)\Box(\langle\!\langle x\rangle\!\rangle(a,y)\mathbf{X}p \wedge \langle\!\langle x\rangle\!\rangle(a,x)\mathbf{X}\neg p)$
- $\theta_1 := \langle\!\langle x\rangle\!\rangle(a,x)\Box(\langle\!\langle x\rangle\!\rangle(a,x)\mathbf{X}(p \wedge \Box(\langle\!\langle x\rangle\!\rangle(a,x)\mathbf{X}p \wedge \langle\!\langle x\rangle\!\rangle(a,x)\mathbf{X}\neg p))$
  $\wedge\langle\!\langle x\rangle\!\rangle(a,x)\mathbf{X}(\neg p \wedge \Box(\langle\!\langle x\rangle\!\rangle(a,x)\mathbf{X}p \wedge \langle\!\langle x\rangle\!\rangle(a,x)\mathbf{X}\neg p)))$
- for every $i \in \mathbb{N}$,
  $\theta i+1 := \theta_i[p \wedge \Box(\langle\!\langle x\rangle\!\rangle(a,x)\mathbf{X}p \wedge \langle\!\langle x\rangle\!\rangle(a,x)\mathbf{X}\neg p \backslash p][\neg p \wedge \Box(\langle\!\langle x\rangle\!\rangle(a,x)\mathbf{X}p \wedge \langle\!\langle x\rangle\!\rangle(a,x)\mathbf{X}\neg p)\backslash\neg p]$

where the notation $\psi_1[\psi_2\backslash\psi_3]$ designates the formula obtained from ψ_1 by replacing any occurrence of subformula ψ_3 in it by ψ_2.

We have that θ_∞ under USL is equivalent to $\{\theta_i\}_{i\in\mathbb{N}}$. Let us suppose that Θ exists and suppose, *w.l.o.g*, it is s.t. the negations in Θ are reduced to atoms (by use of the convenient equivalences in SL). We call a non-trivial universal subformula in Θ a subformula $(a,x)\varphi$ s.t. $\neg\varphi$ is satisfiable and x is universally quantified in Θ. We call Σ_1-SL the set of formulas in SL without non-trivial universal subformula. We call $\mathrm{SL}^1_{\mathrm{R}}$ the fragment of Σ_1-SL without $\mathbf{U}$ nor $\mathbf{R}$ and with $\Box$.

Now, the proof of lemma 4 proceeds by an *absurdum* argument. It is in two parts: first we show that Θ, if it exists, must be equivalent to a formula in $\mathrm{SL}^1_{\mathrm{R}}$. Then we show that it cannot be in $\mathrm{SL}^1_{\mathrm{R}}$.

4.3.1. *Θ is in SL^1_R.* First we prove that Θ, if it exists, is in Σ_1-SL. Let us consider the model $\mathcal{U}$ with domain $\{s_I\}_{I\in\{0,1\}^*}$, with valuation $v(p) = \{s_{I\cdot 1}\}_{I\in\{0,1\}^*}$ and s.t. for every $I \in \{0,1\}^*$, $Ch(a,s_I) = \{\{s_{I\cdot 0}\}\{s_{I\cdot 1}\}\}$. One checks that θ_∞ is true in every states of $\mathcal{U}$. One also checks that for any satisfiable *LTL* formula φ, $\langle\!\langle x\rangle\!\rangle(a,x)\varphi$ holds at any state s_i of $\mathcal{U}$. Then the innermost subformula $(a,x)\varphi$ of Θ s.t. x is universally quantified in Θ is s.t. $\neg\varphi$ is unsatisfiable. So Θ is equivalent to $\Theta[\varphi'[p \vee \neg p\backslash(a,x)\varphi]\backslash\langle\!\langle x\rangle\!\rangle\varphi']$. By iterating this transformation, one eliminates from Θ all its universal quantifiers, so as to obtain a Σ_1-SL formula. Now we can delete the operators $\mathbf{U}$ and $\mathbf{R}$ in Θ.

Lemma 5. *Θ can be written without $\mathbf{U}$ nor $\mathbf{R}$.*

Proof. One observes that Θ is true only in models where a can force the execution to states where she can ensure any satisfiable formula at next state. In particular, if she can ensure $\psi_1\mathbf{U}\psi_2$, ψ_2 is satisfiable and she can ensure it at next state. Formally, Θ is equivalent to the formula Θ' obtained from it by replacing any subformula $\psi = \psi_1\mathbf{U}\psi_2$ by $\bigvee_{0\le k\le|\Theta|}(\mathbf{X}^k\psi_2 \wedge \bigwedge_{0\le i<k}\mathbf{X}^i\psi_1)$, where $\mathbf{X}^k$ stands for a sequence of $\mathbf{X}$*s* of length k. In the introduced subformula, the disjunction $\bigvee_{0\le k\le|\Theta|}$ ensures that a can achieve, in at most $|\Theta|$ transitions, up to $|\Theta|$ possibly contradictory state properties. The deletion of the $\mathbf{R}$ operator proceeds the similar way, by use of the equivalence $\varphi_1\mathbf{R}\varphi_2 \leftrightarrow (\varphi_2\mathbf{U}(\varphi_1 \wedge \varphi_2)) \vee \Box\varphi_1$. □

4.3.2. *Θ' is not in SL^1_R.* To prove that Θ' is not in $\mathrm{SL}^1_{\mathrm{R}}$, we use a compactness argument over formulas $\{\theta_i\}_{i\in\mathbb{N}}$. To proceed so, we give both an axiomatization of *at-most-binary* trees (that are trees in which every node has one or two successors) and a translation from $\mathrm{SL}^1_{\mathrm{R}}$ to Σ^1_1, the fragment of second-order-logic with only existential quantifiers over sets.

First, a model $\mathcal{M}$ being an *at-most-binary tree* rooted in r is axiomatized by:

$$\begin{aligned}\mathrm{BT}(R, R^*, r) := \\ &\forall s(s \neq r \rightarrow (\exists_{=1} s' R(s', s)) \wedge \neg R(s, r) \\ &\wedge (R^*(r, s)) \wedge \exists_{\leq 2} s'(R(s, s'))) \\ &\wedge \forall s, s'(R^*(s, s') \leftrightarrow (\neg R^*(s', s)) \\ &\wedge (R(s, s') \vee \exists s''(R(s, s'') \wedge R^*(s'', s'))))\end{aligned}$$

The translation uses the property, for a set of states S, to define a strategy from s. It means that S is a sub-tree in $\mathcal{M}$ rooted in s. It is axiomatized the following way :

$$\begin{aligned}\mathit{Strat}(S, s) := S(s) \wedge \forall s' \neq s(S(s') \leftrightarrow \\ &(R^*(s, s') \wedge \exists s''(S(s'') \wedge R(s', s'')) \\ &\wedge \forall s''(R^*(s, s'') \wedge R^*(s'', s') \rightarrow S(s'')))\end{aligned}$$

Now we can translate Σ_1-SL into Σ^1_1 under *at-most-binary* trees:

$$\begin{array}{lcl}
[q]_S(s) & := & q(s) \text{(for every atom } q) \\
[\varphi_1 \vee \varphi_2]_S(s) & := & [\varphi_1]_S(s) \vee [\varphi_2]_S(s) \\
[\varphi_1(s) \wedge \varphi_2]_S(s) & := & [\varphi_1]_S(s) \wedge [\varphi_2]_S(s) \\
[\neg\varphi]_S(s) & := & \neg[\varphi]_S(s) \\
[(a, x)\varphi]_S(s) & := & [\varphi]_{S_x}(s) \\
[\langle\!\langle x \rangle\!\rangle \varphi]_S(s) & := & \exists S_x(\mathit{Strat}(S_x, s) \wedge [\varphi]_S(s)) \\
[\mathbf{X}\varphi]_S(s) & := & \forall s'((S(s') \wedge S(s, s')) \rightarrow [\varphi_S](s')) \\
[\Box\varphi]_S(s) & := & \forall s'((S(s') \wedge S^*(s, s')) \rightarrow [\varphi_S](s'))
\end{array}$$

Let us notice that if φ is a sentence, then the translated $[\varphi]_S(s)$ does not depend on S, so that we can consider the formula with one free variable of state $[\varphi](s)$.

Then, each θ_i is equivalent to a formula $\varphi_i := \exists S_{x_1}, \ldots \exists S_{X_n} \phi_i(X_1, \ldots X_n)$ where $\phi_i(X_1, \ldots X_n)$ is a first-order formula. So φ_i is satisfiable if and only if $\phi_i(X_1, \ldots X_n)$ is. Since the SAT-problem for Σ^1_1 reduces to the SAT problem for first-order logic, we can apply the compactness theorem: for every $i \in \mathbb{N}$, consider the following class of models T_i: up to rank i, each member is the binary tree with left direction going to p and right direction going to $\neg p$, and each state loops on itself. After rank i each node has a single successor. Each transition from state s to s' corresponds to a choice for a in s. For each i, θ_i is true in every member of T_i. Now let I be a finite set of indexes. For any $i \in I$, θ_i is true in any member of $T_{max(I)+1}$ and θ_∞ is false in some of these models. If Θ' is in Σ_1-SL then, for any finite $\{\theta_i\}_{i\in I} \subset \{\theta_i\}_{i\in\mathbb{N}}$, $\{\mathrm{BT}(R, R^*, r)\} \cup \{\theta_i\}_{i\in I} \cup \{\neg\Theta'\}$ is satisfiable. Then $\{\mathrm{BT}(R, R^*, r)\} \cup \{\theta_i\}_{i\in\mathbb{N}} \cup \{\neg\Theta'\}$ is satisfiable, which is a contradiction. So Θ' is not SAT-equivalent to a first-order formula, it is not in $\mathrm{SL}^1_{\mathrm{R}}$.

In conclusion, if Θ' exists, it must and cannot be in $\mathrm{SL}^1_{\mathrm{R}}$. Then Θ' does not exist, neither does Θ, and the formula θ_∞ is not expressible in SL, which achieves the proof of theorem 4.

5. Model-checking

In this section we discuss the model checking of USL and USL$^{\circ}$. The model-checking problem for USL has the same complexity as for SL and ATL_{sc}, which are non-elementarily decidable. Nevertheless, the problem for USL$^{\circ}$ is much more tractable, since we get a PSPACE completeness result, similar to that for ATL_{sc} under memoryless strategies.

5.1. **USL.** The model-checking for USL with full memory strategies is non-elementarily decidable. The embedding of SL into USL provides the hardness part. The upper bound requires the effective construction of an algorithm to perform model-checking.

Theorem 2. *Let us call USL[k-alt] the set of USL formulas with at most k quantifier alternations, then the model-checking problem for USL[k-alt] is k-**EXPSPACE** hard.*

Proof. Let $k \in \mathbb{N}$ and let MC-SL[k-alt]$(\mathcal{M}, s, \varphi)$ be the problem of deciding the truth of an SL [k-alt] sentence φ at state s of a *CGS* $\mathcal{M}$. It is $k-$ **EXPSPACE**-hard. By theorem 1 it reduces to the problem MC-USL $(\mathcal{G}_{\mathcal{A}}, s, \varphi_{\text{USL}})$. The transformation preserves the number of quantifier alternations. It also brings a model with domain of size $|\mathcal{M}| \times X^{\Sigma}$. And it brings a formula φ' with size bounded by $3 \times |\varphi|$. So it is in linear space with regard to the size of φ and it does not affect the k-**EXPSPACE** lower bound for its model-checking. □

The effective existence of a non-elementary algorithm for the model-checking of USL is obtained by construction of a non-deterministic parity automaton $\mathcal{N}_{\varphi}^{\mathcal{M},s}$ for any formula φ, *NATS* $\mathcal{M}$, and state s of $\mathcal{M}$. This automaton recognizes a language $\mathcal{L}(\mathcal{N}_{\varphi}^{\mathcal{M},s})$ encoding plans π with empty contexts s.t. $\mathcal{M}, \pi, s \models \varphi$. In case φ is a sentence, $\mathcal{M}, \pi, s \models \varphi$ is equivalent to the emptiness of $\mathcal{L}(\mathcal{N}_{\varphi}^{\mathcal{M},s})$ The proof is adapted from [MMPV11, DLM10, DCL11].

Let us first give the necessary preliminary definitions about trees and automata:

5.1.1. *Trees.*

Definition 12. *Let Δ and S be two finite sets. A Δ-labelled S-tree is a pair $\mathcal{T} = \langle T, l\rangle$ where:*

- *$T \subseteq S^*$ is a non empty set of finite words upon alphabet S, satisfying the following property: for every non empty word $n = m.s \in T$ s.t. $m \in S^*$ and $s \in S$, we have that $m \in T$*
- *$l : T \to \Delta$ is a labelling function*

Let $\mathcal{T} = \langle T, l\rangle$ be such a tree. Let $n \in T$, the set of directions in T from n is the set $\text{dir}_n(T) = \{s \in S \mid n.s \in T\}$. The set of infinite paths of $\mathcal{T}$ is the set $path_{\mathcal{T}} = \{s_0.s_1 \cdots \in S^{\omega} \mid \forall i \in \mathbb{N}, s_0.s_1 \dots s_i \in T\}$. Let $\rho = (s_i)_{i \in N}$, then $l(\rho)$ denotes the infinite sequence $(l(s_i))_{i \in \mathbb{N}}$, and $\text{Inf}(l(\rho))$ is the set of characters in Δ appearing infinitely often in $l(\rho)$.

Now, let $\Delta = \Delta_1 \times \Delta_2$, and let $\mathcal{T} = \langle T, l\rangle$ be a Δ-labelled S-tree. Then for $n \in T$, we write $l(n) = (l_1(n), l_2(n))$ with $l_1(n) \in \Delta_1$ and $l_2(n) \in \Delta_2$. Furthermore, for $i \in \{1, 2\}$, we denote by $\text{Proj}_{\Delta_i}(\mathcal{T})$ the Δ_i-labelled tree $\mathcal{T}_i = \langle T, l_i\rangle$.

5.1.2. *Alternating-tree automata.* Alternating-tree automata are a generalization of nondeterministic tree automata (their closure under complementation). The definition of the transition function for these automata previously requires the definition of *positive boolean formulas*:

Definition 13. *Let P be a set of atomic propositions, the set of positive boolean formulas upon P (PBF(P)) is generated by the following grammar:*

$$\varphi := p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \top \mid \bot$$

where $p \in P$

The satisfaction of a positive boolean formula is defined recursively in the common way, under a valuation $v : P \to \{\top, \bot\}$. We write that a subset P' of P satisfies a positive boolean formula φ iff the valuation $v_{P'}$ defined by $v_{P'}(p) = \top$ iff $p \in P'$ does.

Definition 14. *Let S and Δ be two finite sets. An Alternating S-automaton upon Δ ($\langle S, \Delta\rangle -$ ATA for short), is a $4-$tuple $\mathcal{A} = \langle Q, q_0, \tau, Acc\rangle$ where:*

- *Q is a finite set of states,*

- $q_o \in Q$ *is the initial state,*
- $\tau : (Q \times \Delta) \rightarrow PBF(S \times Q)$ *is the transition function,*
- $Acc : Q^\omega \rightarrow \{\top, \bot\}$ *is an* acceptation function.

Under these notations we call S the set of directions *of automaton $\mathcal{A}$. Non-deterministic S-automata upon Δ ($\langle S, \Delta\rangle - NTAs$ for short) can be defined as particular cases of $\langle S, \Delta\rangle - ATAs$: a NTA is an ATA in which each conjunction in the transition function (written in disjunctive normal form) τ has exactly one move associated with each direction in S. Formally, an NTA is an ATA in which for all state q and for every letter $d \in \Delta$, the transition $\tau(q, d)$ has shape:*

$$\tau(q, d) = \bigvee_{i \in I(q)} \left(\bigwedge_{s \in S} (s, q_{i,s}) \right)$$

where $I(q)$ is a finite set of indices and the $q_{i,s}$ are states in Q.

5.1.3. *Runs and parity acceptance conditions.*

Definition 15. *Let $\mathcal{A} = \langle Q, q_o, \tau, Acc\rangle$ be an $\langle S, \Delta\rangle - ATA$, and $\mathcal{T} = \langle T, l\rangle$ a Δ-labelled S-tree. A* run *of $\mathcal{A}$ on $\mathcal{T}$ is a $S \times Q$-tree $\mathcal{U} = \langle U, p\rangle$ s.t. for every node $u \in U$ s.t. $u = (t, q) = (t_o t_1 \dots t_n, q_o q_1 \dots q_n)$ we have that:*

- $t \in T$
- *The set $dir_u(U) = \{(s_o, q'_o), (s_1, q'_1), \dots, (s_n, q'_n)\} \subseteq S \times Q$ satisfies $\tau(q, p(t))$.*

A run $\mathcal{U}$ is accepting *if $Acc(v) = \top$ for every infinite path $v \in (S \times Q)^\omega$ in $path_{\mathcal{U}}$. A tree $\mathcal{T}$ is accepted by $\mathcal{A}$ iff there is an accepting run of $\mathcal{A}$ upon $\mathcal{T}$.*

In the remaining of this proof we use *parity* acceptance conditions. A parity acceptance condition for automaton $\mathcal{A}$ is identified by a chain of subsets included in the set Q of states in $\mathcal{A}$: $F_1 \subseteq \dots \subseteq F_k = Q$ where $k \in \mathbb{N}$. The acceptance condition is given by $Acc(v) = \top$ iff:

$$\min(\{l \leq k \mid F_k \cap \mathrm{Inf}(\mathrm{Proj}_Q(v)) \text{ is even})\}$$

Under these notations, number k is called the index of the automaton. For automaton $\mathcal{A}$ it is denoted by id_A. And the size of $\mathcal{A}$, $|\mathcal{A}|$ is given by the number of its states. An *ATA* with such acceptation condition is called an alternating-parity tree automaton, that we abbreviate by *APT*. Similarly, an *NTA* with a parity acceptance condition is an *NPT*.

5.1.4. *Preliminary lemmas.* The remaining of the proof mainly consists in building, for every *NATS* $\mathcal{M}$, state s in $\mathcal{M}$ and formula $\varphi \in$ USL, an *APT* $\mathcal{A}^{\mathcal{M}_\varphi}$ accepting exactly the encodings of plans π s.t. $\mathcal{M}, \pi, s \models \varphi$. The used encoding is formally defined Def.16. $\mathcal{A}^{\mathcal{M}_\varphi}$ is built in induction upon φ complexity. We first give the different lemmas that are used to pass the inductive steps in this construction.

Basically, the steps for boolean operators correspond to the intersection and complementation operation for *APTs*. The case for temporal operators **X** and **U** are treated the usual way for temporal logics.

The case for existential operators uses the operation of existential projection for *NTAs*. Then, passing an existential step requires the nondeterminization of the *APT*. A lemma for nondeterminization of *APTs* will be needed at this point. It is given as lemma 7. The building of the *APT* for φ goes through an alternation between *APTs* for treating the complementation cases and *NPTs* for treating the existential projection.

Let us first give the needed lemmas and their references:

Lemma 6 (Intersection and complementation)**.** *[MS87,MS95] Let $\mathcal{A}$ be an $\langle S, \Delta\rangle - APT$ accepting language A and $\mathcal{B}$ be a $\langle S, \Delta\rangle$-APT accepting language B.*

- *There is an $\langle S, \Delta\rangle - APT$ C accepting language $A \cap B$. The size $|C|$ of C is bounded by $|\mathcal{A}| + |\mathcal{B}|$ and its index is bounded by $max(id_A, id_B)$.*

- *There is an $\langle S, \Delta\rangle - APT$ $\mathcal{D}$ accepting language $\overline{A}$, the complementary of language A. Its size and index are the same as those of $\mathcal{A}$. More precisely, if $\mathcal{A} = \langle Q, q_o, \tau, Acc\rangle$, then $\mathcal{D} = \langle Q, q_o, \overline{\tau}, \overline{Acc}\rangle$ where for every $(q, d) \in Q \times \Delta$, $\overline{\tau}(q, d) = \neg\tau(q, d)$ and $\overline{Acc}$ is the complementary condition of Acc.*

Lemma 7 (Nondeterminization). *[MS95] Let $\mathcal{A}$ be an $\langle S, \Delta\rangle - APT$. There is a $\langle S, \Delta\rangle - NPT$ $\mathcal{A}$ accepting the same language as $\mathcal{A}$, and s.t. $|\mathcal{N}| = 2^{\bigcirc(|\mathcal{A}|.id_{\mathcal{A}}.log(|\mathcal{A}|))}$ and $id_{\mathcal{N}} = \bigcirc(|\mathcal{A}|.id_{\mathcal{A}}.log(|\mathcal{A}|))$.*

5.1.5. *Inputs of the automata.* The inputs for the automata checking the satisfaction of formulas by a plan are composed by an encoding of a plan together with a state data. From Def. 12, the input must be from a finite set. However, USL semantics defines choices by help of a plan $\pi = (\mu, \kappa)$, where context κ stands for any word upon $(\Sigma \times X)$. So its domain is infinite.

Nevertheless, the context registers data only about the previously evaluated binders. Then its size is actually bound by the maximal number of binders in the formula under evaluation, so that its domain can be made finite. Let φ be an USL formula and let ψ be one of its subformulas. We write $\mathrm{bd}(\psi;\varphi)$ the *binder depth* of ψ in φ. The notion is defined that way: $\mathrm{bd}(\psi;\varphi)$ is the number of φ's subformulas of type $\theta := (A \triangleright x)\theta'$ s.t. ψ is a subformula of θ. Now, let K^k be the set of finite words over $(\Sigma \times X)^*$ of length k. Then the definition of our automaton uses $K^{\mathrm{bd}(\psi;\varphi)}$.

Furthermore, let s be a state in a *NATS*. We call a *decision* from s (Dc^s) a function from Σ to $\mathcal{P}(M)$ s.t. for every $a \in \Sigma$, $Dc^s(a) \in Ch(a, s)$.

We write DC^s the set of decisions from s. We want to use it as an uniform set over the different states. So we need to delete the parameter s in the writing of DC^s: for every $s \in M$, we enumerate by $[1, \ldots, d_s]$ the different decisions from s. And we note e_s the corresponding function from $[1, \ldots, d_s]$ to DC^s. Let $d_{\max}$ be the maximal such d_s for $s \in M$. Then, for every $s \in M$, we complete e_s s.t. for every $d_s \leq i \leq d_{\max}$, $e_s(i) = e_s(d_s)$. We designate by D the set $n \in [0, \ldots d_{\max}]$. Now, let us write Ac and call *actions* the set of partial functions from X to D.

The input of trees we examine are taken in $S \times Ac \times K^{\mathrm{bd}(\psi;\varphi)}$. Let $(s, \alpha, \kappa) \in S \times Ac \times K^{\mathrm{bd}(\psi;\varphi)}$, it defines as its outcome an unique $out(s, \alpha, \kappa) \subseteq M$: $out(s, \alpha, (A, x)) = \bigcap_{a\in A} e_s(\alpha(x))(a)$ and $out(s, \alpha, \kappa \cdot (A, x)) = out(s, \alpha, \kappa) \cap \bigcap_{a\in A} e_s(\alpha(x))(a)$ if it is not empty, else $out(s, \alpha, \kappa)$.

Before coming to the effective building of the *APT*, we define a *state-plan Encoding*:

Definition 16. *Let $\mathcal{M}$ be a NATS, s a state in M, π a plan for $\mathcal{M}$ and $j \in \mathbb{N}$. Then a $M \times Ac \times K^j$-labelled $\mathcal{M}$-tree $\mathcal{T} = \langle T, l\rangle$, where $T \subseteq track_{\mathcal{M}}$, is the* state-plan Encoding *for $\pi = (\mu, \kappa)$ iff it holds that for every $t \in T$, $l(t) = (last(t), \alpha, \kappa)$, where α is s.t. $e_{last(t)}(\alpha) = \{\mu(a, t)\}_{a\in\Sigma}$. If T is the plan-state encoding for π, we call $Proj_{Ac\times K^j}(\mathcal{T})$ the plan encoding for π.*

5.1.6. *Checking a state-plan encoding.* Now we can give the main step lemma of that proof. It states that given a *NATS* $\mathcal{M}$ and a formula φ in USL, one can build an automaton accepting exactly those trees that are state-plan encodings of plans satisfying φ in $\mathcal{M}$.

Lemma 8. *Let $\mathcal{M}$ be a NATS with domain M and φ an USL formula. Then, there is an $\langle M, M \times Ac \times K^o\rangle - APT$ $\mathcal{A}^{\mathcal{M}}_{\varphi}$ s.t. for all states s of $\mathcal{M}$ and plans π with context $\in K^o$ for $\mathcal{M}$, it holds that $\mathcal{M}, \pi, s \models \varphi$ iff $\mathcal{T} \in \mathcal{L}(\mathcal{A}^{\mathcal{M}}_{\varphi})$, where $\mathcal{T}$ is the state-plan encoding for π.*

Proof. The proof of the lemma is led by effective building of $\mathcal{A}^{\mathcal{M}}_{\varphi}$. The building is inductive upon φ complexity and the induction hypothesis is: for every subformula ψ of φ, there is an $\langle M, M \times Ac \times K^{\mathrm{bd}(\psi;\varphi)}\rangle - APT$ $\mathcal{A}^{\mathcal{M}}_{\psi}$ s.t. for every state s of $\mathcal{M}$ and plan $\pi \in K^{\mathrm{bd}(\psi;\varphi)}$, it holds that $\mathcal{M}, \pi, s \models \psi$ iff $\mathcal{T} \in \mathcal{L}(\mathcal{A}^{\mathcal{M}}_{\psi})$, where $\mathcal{T}$ is the state-plan encoding for π.

In the following we write Δ for $M \times Ac \times K^{\mathrm{bd}(\psi;\varphi)}$.

- Case ψ is an atomic proposition: the only thing to check is that the state at the root of the tree given in entry satisfies ψ. Then $\mathcal{A}^{\mathcal{M}}_{\psi} = \langle\{\overline{\psi}\}, \overline{\psi}, \tau_{\psi}, (\{\}\{\psi\})\rangle$ s.t. for every $ent \in \Delta$, $\tau_{\psi}(\overline{\psi}, ent) = \top$ if $\psi \in \nu(s)$ and $\tau_{\psi}(\overline{\psi}, (P, s)) = \bot$ otherwise.

- Case $\psi = \neg\psi_1$. If $\mathcal{A}^{\mathcal{M}}_{\psi_1} = \langle Q_{\psi_1}, q_{o_{\psi_1}}, \tau_{\psi_1}, Acc_{\psi_1}\rangle$, then by lemma 6, $\mathcal{A}^{\mathcal{M}}_{\psi} = \langle Q_{\psi_1}, q_{o_{\psi_1}}, \overline{\tau_{\psi_1}}, \overline{Acc_{\psi_1}}\rangle$.
- Case $\psi = \psi_1 \wedge \psi_2$. If $\mathcal{A}^{\mathcal{M}}_{\psi_1} = \langle Q_{\psi_1}, q_{o_{\psi_1}}, \tau_{\psi_1}, Acc_{\psi_1}\rangle$ and $\mathcal{A}^{\mathcal{M}}_{\psi_2} = \langle Q_{\psi_2}, q_{o_{\psi_2}}, \tau_{\psi_2}, Acc_{\psi_2}\rangle$ then $\mathcal{A}^{\mathcal{M}}_{\psi} = \langle Q_{\psi}, q_o, \tau_{\psi}, Acc_{\psi}\rangle$ where:
  - $Q_{\psi} = \{q_{o_{\psi}}\} \cup Q_{\psi_1} \cup Q_{\psi_2}$ and $q_o \notin Q_{\psi_1} \cup Q_{\psi_2}$
  - for every $ent \in \Delta$, $\tau_{\psi}(q_{o_{\psi}}, ent) = \tau_{\psi_1}(q_{o_{\psi_1}}, ent) \wedge \tau_{\psi_2}(q_{o_{\psi_2}}, ent)$
  - for every $q \in Q_{\psi_1} \cup Q_{\psi_1}$ and for every $ent \in \Delta$, $\tau_{\psi}(q, ent) =$
    - $\tau_{\psi_1}(q, ent)$ if $q \in Q_{\psi_1}$
    - $\tau_{\psi_2}(q, ent)$ if $q \in Q_{\psi_2}$
  - If $Acc_{\psi_1} = (F_1^{\psi_1}, \dots F_{k_1}^{\psi_1})$ and $Acc_{\psi_2} = (F_1^{\psi_2}, \dots F_{k_2}^{\psi_2})$, $\max(\{k_1, k_2\}) = k_i$ and $\min(\{k_1, k_2\}) = k_j$ then $Acc_{\psi} = (F_1^{\psi_1} \cup F_1^{\psi_2}, \dots, F_{k_j}^{\psi_1} \cup F_{k_j}^{\psi_2}, F_{k_j+1}^{\psi_i}, \dots F_{k_i-1}^{\psi_i}, Q_{\psi})$.
- The resolution of case $\psi = \mathbf{X}\psi_1$ consists in a run of the automaton for ψ on the successors of the root of the tree given in entrance. These successors are given by the outcomes of the label at the root. Concretely, if $\mathcal{A}^{\mathcal{M}}_{\psi_1} = \langle Q_{\psi_1}, q_{o_{\psi_1}}, \tau_{\psi_1}, Acc_{\psi_1}\rangle$ then $\mathcal{A}^{\mathcal{M}}_{\psi} = \langle Q_{\psi}, q_{o_{\psi}}, \tau_{\psi}, Acc_{\psi}\rangle$ where:
  - $Q_{\psi} = \{q_{o_{\psi}}\} \cup Q_{\psi_1}$ and $q_o \notin Q_{\psi_1}$
  - for every $ent \in \Delta$, $\tau_{\psi}(q_{o_{\psi}}, ent) = \bigwedge_{s \in out(ent)}(s, q_{o_{\psi}})$
  - for every $q \in Q_{\psi} \neq q_{o_{\psi}}$ and for every $ent \in \Delta$, $\tau_{\psi}(q, ent) = \tau_{\psi_1}(q, ent)$
  - If $Acc_{\psi_1} = (F_1^{\psi_1}, \dots F_{k_1}^{\psi_1})$ then $Acc_{\psi} = (F_1^{\psi_1}, \dots F_{k_1}^{\psi_1} \cup \{q_{o_{\psi}}\})$
- The case $\psi = \psi_1 \mathbf{U} \psi_2$ is resolved by use of the equivalence $(\psi_1 \mathbf{U} \psi_2) \leftrightarrow \psi_1 \vee (\psi_2 \wedge \mathbf{X}(\psi_1 \mathbf{U} \psi_2))$. Given that automata for ψ_1 and ψ_2 are defined, automaton for ψ returns the boolean combination by help of function τ, and the subformula $\mathbf{X}(\psi_1 \mathbf{U} \psi_2))$ induces the initial state looping on itself. To prevent the run from looping indefinitely on the initial state $q_{o_{\psi}}$, $q_{o_{\psi}}$ is included in the first set of the parity acceptance condition. Precisely, if $\mathcal{A}^{\mathcal{M}}_{\psi_1} = \langle Q_{\psi_1}, q_{o_{\psi_1}}, \tau_{\psi_1}, Acc_{\psi_1}\rangle$ and $\mathcal{A}^{\mathcal{M}}_{\psi_2} = \langle Q_{\psi_2}, q_{o_{\psi_2}}, \tau_{\psi_2}, Acc_{\psi_2}\rangle$ then $\mathcal{A}^{\mathcal{M}}_{\psi} = \langle Q_{\psi}, q_o, \tau_{\psi}, Acc_{\psi}\rangle$ where:
  - $Q_{\psi} = \{q_{o_{\psi}}\} \cup Q_{\psi_1} \cup Q_{\psi_2}$ and $q_o \notin Q_{\psi_1} \cup Q_{\psi_2}$
  - for every $ent \in \Delta$, $\tau_{\psi}(q_{o_{\psi}}, ent) = \tau_{\psi_2}(q_{o_{\psi_2}}, ent) \vee (\tau_{\psi_1}(q_{o_{\psi_1}}, ent \wedge \bigwedge_{s \in out(ent)}(s, q_{o_{\psi}}))$
  - for every $q \in Q_{\psi_1} \cup Q_{\psi_2}$ and for every $ent \in \Delta$, $\tau_{\psi}(q, ent) =$
    - $\tau_{\psi_1}(q, ent)$ if $q \in Q_{\psi_1}$
    - $\tau_{\psi_2}(q, ent)$ if $q \in Q_{\psi_2}$
  - If $Acc_{\psi_1} = (F_1^{\psi_1}, \dots F_{k_1}^{\psi_1})$ and $Acc_{\psi_2} = (F_1^{\psi_2}, \dots F_{k_2}^{\psi_2})$, $\max(\{k_1, k_2\}) = k_i$ and $\min(\{k_1, k_2\}) = k_j$ then $Acc_{\psi} = (\{q_{o_{\psi}}\} \cup F_1^{\psi_1} \cup F_1^{\psi_2}, \dots, \{q_{o_{\psi}}\} \cup F_{k_j}^{\psi_1} \cup F_{k_j}^{\psi_2}, \{q_{o_{\psi}}\} \cup F_{k_j+1}^{\psi_i}, \dots \{q_{o_{\psi}}\} \cup F_{k_i-1}^{\psi_i}, Q_{\psi})$.
- The case for $\psi = (A \triangleright x)\psi_1$ only consists in a transformation of the transition function, so that it is equal to the transition in automaton for ψ_1 in which entry the choices along x made by A would be added. If $\mathcal{A}^{\mathcal{M}}_{\psi_1} = \langle Q_{\psi_1}, q_{o_{\psi_1}}, \tau_{\psi_1}, Acc_{\psi_1}\rangle$, then $\mathcal{A}^{\mathcal{M}}_{\psi} = \langle Q_{\psi_1}, q_{o_{\psi_1}}, \tau_{\psi}, Acc_{\psi_1}\rangle$ where for every $q \in Q_{\psi_1}$ and for every $(s, \alpha, \mu) \in \Delta$, $\tau_{\psi_1}(q, (s, \alpha, \kappa)) = \tau_{\psi}(q, (s, \alpha, \kappa[A \rightarrow x])))$.
- The case for $\psi = (A \not\triangleright x)\psi_1$ again only consists in a transformation of the transition function. It is so that the new transition function is equal to the transition in automaton for ψ_1 in which entry the choices along x made by A would be deleted. If $\mathcal{A}^{\mathcal{M}}_{\psi_1} = \langle Q_{\psi_1}, q_{o_{\psi_1}}, \tau_{\psi_1}, Acc_{\psi_1}\rangle$, then $\mathcal{A}^{\mathcal{M}}_{\psi} = \langle Q_{\psi_1}, q_{o_{\psi_1}}, \tau_{\psi}, Acc_{\psi_1}\rangle$ where, for every $q \in Q_{\psi_1}$ and for every $(s, \alpha, \kappa) \in \Delta$, $\tau_{\psi_1}(q, (s, \alpha, \kappa)) = \tau_{\psi}(q, (s, \alpha, \kappa[A \nrightarrow x]))$.
- For the case $\psi = \langle\!\langle x \rangle\!\rangle \psi$, the transition function of the automaton for ψ has gives the disjunction of all possible transitions corresponding to each decision given by possibles strategy instantiating x after each track. This operation is performed for *NPTs*, so that we need first to nondeterminize the automaton for ψ_1. If $\mathcal{A}^{\mathcal{M}}_{\psi_1} = \langle Q_{\psi_1}, q_{o_{\psi_1}}, \tau_{\psi_1}, Acc_{\psi_1}\rangle$ is an $\langle M, \Delta\rangle - APT$, then by lemma 7, there is an $\langle M, \Delta\rangle - NPT$ $\mathcal{A}'^{\mathcal{M}}_{\psi_1} = \langle Q'_{\psi_1}, q'_{o_{\psi_1}}, \tau'_{\psi_1}, Acc'_{\psi_1}\rangle$ accepting the same language, and s.t. $|\mathcal{A}'^{\mathcal{M}}_{\psi_1}| = 2^{\bigcirc(|\mathcal{A}^{\mathcal{M}}_{\psi_1}|.id_{\mathcal{A}^{\mathcal{M}}_{\psi_1}}.log(|\mathcal{A}^{\mathcal{M}}_{\psi_1}|))}$ and $id_{\mathcal{A}'^{\mathcal{M}}_{\psi_1}} = \bigcirc(|\mathcal{A}^{\mathcal{M}}_{\psi_1}|.id_{\mathcal{A}^{\mathcal{M}}_{\psi_1}}.log(|\mathcal{A}^{\mathcal{M}}_{\psi_1}|))$. For the

projection we define the $\langle M, \Delta\rangle - NPT \mathcal{A}^{\mathcal{M}}_{\psi} = \langle Q'_{\psi_1}, q'_{o_{\psi_1}}, \tau_{\psi}, Acc'_{\psi_1}\rangle$ where for all $(s, \alpha, \kappa) \in \Delta, \tau_{\psi}(q, (s, \alpha, \kappa)) = \bigvee_{Dc^s \in DC^s} \tau'_{\psi_1}(q, (s, \alpha_{dcs}, \kappa))$, where α_{dcs} is the minimal α s.t. $(e_s(k) = e_s(\alpha)[x \to Dc^s])$.

□

5.1.7. *Conclusion of the proof.* A last lemma is needed for concluding the proof, so as to ensure that the M component of labelling for the nodes of the state-plan encoding is coherent with the node itself:

Lemma 9 (direction projection [MMPV11])**.** *Let $\mathcal{N}$ be an $\langle M, M \times \Delta\rangle$-NPT and $s_o \in M$. Then there is an $\langle M, \Delta\rangle$-NPT $\mathcal{N}^{s_o}$ s.t. for every Δ-labelled M-tree $\mathcal{T} = \langle T, v\rangle$, $\mathcal{T} \in \mathcal{L}(\mathcal{N}^{s_o})$ iff $\mathcal{T}' \in \mathcal{L}(\mathcal{N})$, where $\mathcal{T}' = \langle T', v'\rangle$ is the $M \times \Delta$-labelled M-tree s.t. $v'(t) = (v(t), last(s_O.t))$, for every $t \in T$. Moreover, $|\mathcal{N}^{s_o}| = |\Delta|.|\mathcal{N}|$ and $id_{\mathcal{N}^{s_o}} = id_{\mathcal{N}}$.*

Now we can prove the effective non-elementary decidability of the model-checking for USL:

Theorem 3. *The model-checking problem for USL can be run in **NONELEMENTARY** with regard to the size of the formula.*

Proof. As seen in lemma 8, for any *NATS* $\mathcal{M}$, state $s \in M$ and plan π for $\mathcal{M}$, the problem MC-USL $(\mathcal{M}, s, \pi, \varphi)$ of model-checking formula φ in state s and plan π for $\mathcal{M}$ reduces to the question whether $\mathcal{T} \in \mathcal{L}(\mathcal{A}^{\mathcal{M}}_{\varphi})$, where $\mathcal{T}$ is the state-plan encoding for π. Furthermore, $\mathcal{A}^{\mathcal{M}}_{\varphi}$ is a $\langle M, M \times Ac \times K^o\rangle - APT$. By lemma 7, there is a $\langle M, Ac \times K^o\rangle - NPT$ $\mathcal{N}^{\mathcal{M}}_{\varphi}$ accepting the same language as $\mathcal{A}^{\mathcal{M}}_{\varphi}$, and s.t. $|\mathcal{N}^{\mathcal{M}}_{\varphi}| = 2^{\bigcirc(|\mathcal{A}^{\mathcal{M}}_{\varphi}|.id_{\mathcal{A}^{\mathcal{M}}_{\varphi}}.log(|\mathcal{A}^{\mathcal{M}}_{\varphi}|))}$ and $id_{\mathcal{N}^{\mathcal{M}}_{\varphi}} = \bigcirc(|\mathcal{A}^{\mathcal{M}}_{\varphi}|.id_{\mathcal{A}^{\mathcal{M}}_{\varphi}}.log(|\mathcal{A}^{\mathcal{M}}_{\varphi}|))$.

By lemma 9 applied on $\mathcal{N}^{\mathcal{M}}_{\varphi}$, we have an $\langle M, Ac \times \kappa^o$-$NPT$ $\mathcal{N}^{\mathcal{M},s}_{\varphi}$ accepting exactly encodings of plans π s.t. $\mathcal{M}, \pi, s \models \varphi$.

One easily checks that, φ being a sentence, for every plan π with empty context, $\mathcal{M}, \pi, s \models \varphi$ iff and only $\mathcal{M}, \mu_{\emptyset}, \kappa_{\emptyset}, s \models \varphi$. Then $\mathcal{M}, s \models \varphi$ iff $\mathcal{L}(\mathcal{N}^{\mathcal{M},s}_{\varphi})$ is not empty. The emptiness problem for non-deterministic automata with n states and index k is solvable in $\bigcirc(n^h)$ [KV98]. Let us note $|\varphi|$ the length of formula φ. Each step of building for $\mathcal{N}^{\mathcal{M}}_{\varphi}$ increases the size of the current automaton $\mathcal{M}$ at most from $|\mathcal{M}|$ to $2^{\bigcirc(|\mathcal{M}|id_{\mathcal{M}}.log(|\mathcal{M}|))}$ and its index from $id_{\mathcal{M}}$ to $\bigcirc(|\mathcal{M}|id_{\mathcal{M}}.log(|\mathcal{M}|)$ and there are at most $|\varphi| + 1$ such steps. The step from $\mathcal{N}^{\mathcal{M}}_{\varphi}$ to $\mathcal{N}^{\mathcal{M},s}_{\varphi}$ increases its size by factor $|\mathcal{M}|$. Then $\mathcal{N}^{\mathcal{M},s}_{\varphi}$ has size at most $|\mathcal{A}| \cdot et(m, 2, m)$ and index at most $et(m, 2, m)$, were $m = \bigcirc(|\varphi|.log(|\varphi|)$ and $et(n_1, n_2, n_3)$ is defined, for every $n_2, n_3 \in \mathbb{N}$, by:

- $et(o, n_2, n_3) = n_2$
- for every $n_1 \in \mathbb{N}$, $et(n_1 + 1, n_2, n_3) = n_3^{et(n_1, n_2, n_3)}$

Whence theorem 3, the model-checking for USL is decidable in time $\bigcirc(|\mathcal{M}|.et(m, 2, m))$. □

5.2. **USL^o.** Here we present a result of PSPACE-completeness for the model-checking of USL^o:

Theorem 4. *The model-checking problem for USL^o is PSPACE complete.*

This result reaches the similar result for ATL^o_{sc} [BDCLM09]. The hardness part is inspired by [BDCLM09, BBGK07]. It consists, for any integer k, in the reduction of the satisfiability problem for Quantified Boolean Formulas with k alternations of quantifiers (**QBFSAT**$_l$) into the model-checking for USL^o with memory-less strategies.

We establish the upper bound by giving a PSPACE algorithm for MC-USL^o. It is largely inspired from [BDCLM09]. The leading idea is that the data of a memory-less plan over a *NATS* defines a restriction of that *NATS*, that can be seen itself as a *NATS*. In this model one can use the model checking of LTL for path formulas, provided that its state subformulas are replaced by new atomic propositions.

5.2.1. *MC-USL ^{o} is PSPACE-hard.* This result is obtained by reduction of $\mathbf{QBFSAT}_k$ to MC-USL ^{o}, for every $k \in \mathbb{N}$:

Lemma 10. *For every $k \geq 0$, $\mathbf{QBFSAT}_k$ reduces to MC-USL^{o}.*

Proof. For $k \in \mathbb{N}$, we consider the Σ_k-hard problem $\mathbf{QBFSAT}_k$, that is the satisfiability problem of a formula $\exists X_1 \forall X_2 \exists X_3 \dots Q_k X_k \ \varphi(X_1, X_2, X_3, \dots X_k)$ where Q_k is $\forall$ if k is even and $\exists$ otherwise, and where φ is a boolean formula in Conjunctive Normal Form (CNF) $\varphi = \bigwedge_{j=1,\dots,n} C_j$, so that each C_j is a disjunctive clause on the set of variables $\{X_1, X_2, X_3, \dots, X_k\}$.

From an instance $\mathcal{I}$ of this problem we build the *NATS* $\mathcal{G}_{\mathcal{I}} = \langle Ag_{\mathcal{I}}, M_{\mathcal{I}}, At_{\mathcal{I}}, v_{\mathcal{I}}, Ch_{\mathcal{I}} \rangle$ represented without label on Fig.3: it is turned base, so that the transition from any v_l with $l \geq k$ is decided by agent a_l between $\neg x_{l+1}$ and x_{l+1} and from $\neg x_l$ or x_l it goes necessarily to v_l. Any transition from v_{k+1} loops back to v_{k+1}:

- $M_{\mathcal{I}} = \bigcup_{0 \leq i \leq k} \{x_i, \neg x_i, v_i\} \cup \{v_{k+1}\}$
- For $l, i \in [1, \dots, k]$, $Ch_{\mathcal{I}}(a_i, v_l) =$
  - $\{\{x_l\}\{\neg x_l\}\}$ if $l = i$
  - M if $l \neq i$
- For $l \in [1, \dots, k]$, $Ch_{\mathcal{I}}(a_i, x_l) = Ch_{\mathcal{I}}(a_i, \neg x_l) = \{\{v_{l+1}\}\}$
- $Ch_{\mathcal{I}}(a_i, v_{k+1}) = \{\{v_{k+1}\}\}$

We label the states so as to make true, in each state x_l or $\neg x_l$, the set of clauses $C_j \in \varphi$ compatible with the corresponding truth value for X_l. That is:

- $\forall 1 \leq l \leq k, \mathrm{Lab}(x_l) = \{C_j \mid X_l \in C_j\}$
- $\forall 1 \leq l \leq k, \mathrm{Lab}(\neg x_l) = \{C_j \mid \neg X_l \in C_j\}$

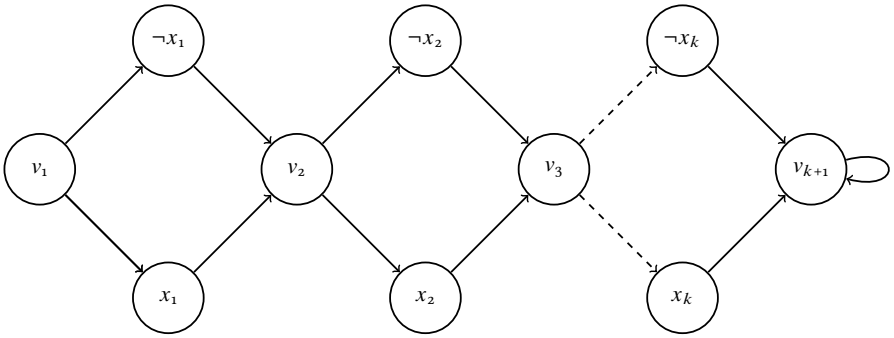

FIGURE 3. $\mathcal{G}_{\mathcal{I}}$: an illustration of $\mathbf{QBFSAT}_k$ as a problem of model- checking for memory-less USL

Since each player a_l efficiently plays once for all and plays by deciding whether X_l holds or not, a strategy for a_l corresponds to a truth value for the variable X_l. Indeed, the formula $\langle\!\langle x_l \rangle\!\rangle (a_l \triangleright x) \Diamond \psi$ (resp. $\neg \langle\!\langle x_l \rangle\!\rangle (a_l \triangleright x) \Diamond \psi$) means that there exists a truth value for X_l s.t. (resp. for every truth value of for X_l,) ψ stands.

Now, consider the following USL formula:

$$\varphi := \langle\!\langle x_1 \rangle\!\rangle (a_1 \triangleright x_1)(\neg \langle\!\langle x_2 \rangle\!\rangle (a_2 \triangleright x_2) \neg (\langle\!\langle x \rangle\!\rangle (a_3 \triangleright x)(\dots (\neg^k \langle\!\langle x_k \rangle\!\rangle (a_k \triangleright x_k) \neg^k (\bigwedge_{j \in 1,\dots,J} \Diamond C_j)) \dots)$$

where for any integer l, $\neg^l$ is equal to $\neg$ if l is odd and empty if l is even. It holds at v_1 iff $\exists X_1 \forall X_2 \exists X_3 \dots Q_k X_k \varphi(X_1, X_2, X_3, \dots X_n)$ is satisfiable, that is if $\mathcal{I}$ is a positive instance of $\mathbf{QBFSAT}_k$. Then for every $k \in \mathbb{N}$, $\mathbf{QBFSAT}_k$ reduces to the model-checking of the fragment of USL^{o} with k

alternations of quantifiers. Because unbounded **QBFSAT** is PSPACE complete then, we get the lower bound for the full model-checking of memory-less USL.

□

5.2.2. *MC-USL $^\circ$ is PSPACE-complete.* The algorithm for MC-USL $^\circ$ proceeds recursively, enumerating strategies in a first time, then using the PSPACE-complete model-checking for LTL in Kripke models.

Let us first define the restriction of a *NATS* by a plan:

Definition 17 ($\mathcal{M}_\pi$). *Let $\mathcal{M} = \langle Ag, M, At, v, Ch\rangle$ be a NATS and let $\pi = (\kappa, \mu)$ be a plan over memory-less strategies in $\mathcal{M}$. Then $\mathcal{M}_\pi = \langle Ag, M, At, v, Ch_\pi\rangle$ is the NATS defined by: for every $(a, s) \in Ag \times M$, $Ch_\pi(a, s) = \{\pi(a, s) \cap c$ iff it is not empty, else $\pi(a, s)\}_{c\in Ch(a,s)}$.*

Now, we introduce the notations enabling to consider a path formula as an LTL formula over its state subformulas. For a formula φ, let us write $Q(\varphi)$ the set of its outermost *properly state subformulas* (*i.e.* of form $\langle\!\langle x\rangle\!\rangle\psi$, $(a \triangleright x)\psi$ or $(a \not\triangleright x)\psi$) and LTLT_φ the formula obtained from φ by replacing in it every subformula ψ in $Q(\varphi)$ by a new atom $\overline{\psi}$.

Let $\mathcal{M}$ be a *NATS* and φ an LTL formula. Then MC-LTL($\mathcal{M}, s, \varphi$) designates the model-checking for universal satisfaction of φ at state s in the Kripke model $\mathcal{M}'$ defined by:

- The domain M and the valuation function v are those of $\mathcal{M}$.
- The transition relation R is: for every $s, s' \in M$, $R(s, s')$ iff $\forall a \in Ag, \exists s \in M, \exists c \in Ch(a, s)$ s.t. $s' \in c$.

With these notations, the procedure is described by Alg.1.

Algorithm 1 MC-USL ($\mathcal{M}, \pi, s, \varphi$)

Require: A *NATS* $\mathcal{M}$, a plan $\pi = (\kappa, \mu)$, $s_o \in M$ and an USL path formula φ
Ensure: YES iff $\mathcal{M}, \pi, s \models \varphi$ $\mathcal{M}' = \mathcal{M}_\pi$

```
for all ψ ∈ Q(φ) do
  for all s' ∈ M do
    if ψ = ⟨⟨x⟩⟩ψ' then
      for all σ ∈ Strat do
        if MC-USL (M', (κ, μ[x → σ]), s', ψ) then
          label s' with ψ̄
        end if
      end for
    else if ψ = (A ▷ x)ψ' then
      if MC-USL (M', (κ[A → x], μ), s, ψ) then
        label s' with ψ̄
      end if
    else if ψ = (A ▷̸ x)ψ' then
      if MC-USL (M', (κ[A ↛ x], μ), s, ψ) then
        label s' with ψ̄
      end if
    end if
  end for
end for
return MC-LTL(M', s, LTLT_φ)
```

Note that a strategy can be stored in space $\bigcirc(|Ag| \times |Q|)$. Since the labelling of the states in M is linear over the size of φ and since the algorithm used as oracle in Alg.1 (MC-LTL) is PSPACE-complete, we have that Alg.1 runs in PSPACE.

6. Related works

Several directions have already been explored for extensions of ATL-ATL*. The work presented here deeply refers to SL [MMV10].

CTL.STIT [Bro10] takes inspiration from *stit*-framework, which originates in philosophy, in order to extend ATL with expression of what agents do in addition to what they can ensure. The aim is to improve ATL expressiveness in terms of verification properties. In particular, it allows assumption-guarantee reasoning. However, the reasoning about strategies and their composition is not improved.

Other formalims have been presented, such as Alternating-Time Temporal Logic with Explicit Strategies (ATLES [WvdHW07]). Rather quantified strategy variables, In ATLES syntax, the ATL operator $\langle\!\langle \cdot \rangle\!\rangle$ is replaced with an operator $\langle\!\langle \cdot \rangle\!\rangle_\rho$ that takes an unquantified strategy term ρ as parameter. ATLES has a **PTIME** model-checking. However, the reference in the syntax to an effective strategy, which is a semantic object, is not convenient in practice.

As in the present work, the revocation of strategies is questioned in [ÅGJ07]. In their proposition, the authors distinguish between revocable strategies as they are defined in SL and ATL, and irrevocable strategies. We believe that strategies in USL offer an adequate synthesis between both views, because they can be later modified and at the same time hold some definitive commitments from the agents.

The idea of unbinding agents from their current strategies is also present in ATL$_{sc}$ [DCL11,BDCLM09] with the operator $\cdot\rangle A\langle\cdot$. Yet, strategies are also automatically revoked in case a given agent is bound to several strategies: it is not possible for an agent to refine its strategy.

7. Conclusion

In this article we defined the logic USL, in which we can reason about agents refining or revoking their strategies. This unifies a rich composition of strategies that allows strategies refinement with the usual revocation of strategies developed in the literature. USL strictly extends SL, and is in particular able to express what we called sustainable capabilities. USL syntax is also more flexible than SL syntax, and is better adapted to modular specifications. Its model checking problem is **NONELEMENTARY** but for its memory-less restriction the model checking is PSPACE-complete.

As future work, we plan to study applications of USL. In particular, as we already noticed in [CBC11], temporal multi-agents logics can be useful to investigate requirement engineering problems. We need to investigate on the possibilities USL offers in the validation of requirement engineering models.

We also want to study further the expressiveness offered by unbinders. In this article we exclusively discussed the meaning of unbinding an agent before she is herself bound again. The expressiveness given by unbinding some agents before binding others should also be analyzed.

Another aspect we want to study is the comparison of USL with the logic QD_μ [Pin07]. This formalism enables to express fixed-point properties about strategies and subsumes SL. Since the concept of sustainable capabilities, which seems to characterize well USL expressiveness, is close to a fix point, we think that this comparison is interesting to investigate.

References

[ÅGJ07] Thomas Ågotnes, Valentin Goranko, and Wojciech Jamroga. Alternating-time temporal logics with irrevocable strategies. In *Theoretical aspects of rationality and knowledge*, pages 15–24, 2007.

[AHK02] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[BBGK07] C. Baier, T. Brazdil, M. Grosser, and A. Kucera. Stochastic game logic. In *Quantitative Evaluation of Systems.*, pages 227 –236, 2007.

[BDCLM09] T. Brihaye, A. Da Costa, F. Laroussinie, and N. Markey. ATL with strategy contexts and bounded memory. *Logical Foundations of Computer Science*, pages 92–106, 2009.

[Bro10] J.M. Broersen. CTL.STIT: enhancing ATL to express important multi-agent system verification properties. In *Proc. of the ninth international joint conference on Autonomous agents and multiagent systems (AAMAS 2010)*, pages 683–690, New York, NY, USA, 2010. ACM.

[CBC11] Christophe Chareton, Julien Brunel, and David Chemouil. A formal treatment of agents, goals and operations using alternating-time temporal logic. In *SBMF*, pages 188–203, 2011.

[CHP10] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Strategy logic. *Inf. & Comp.*, 208(6):677–693, 2010.

[DCL11] Arnaud Da Costa Lopes. *Propriétés de jeux multi-agents*. Phd thesis, École normale supérieure de Cachan, September 2011.

[DLM10] Arnaud Da Costa, François Laroussinie, and Nicolas Markey. ATL with strategy contexts: Expressiveness and model checking. In *IARCS Annual Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 8, pages 120–132, 2010.

[KV98] Orna Kupferman and Moshe Y. Vardi. Weak alternating automata and tree automata emptiness. In *ACM Symposium on Theory of computing*, pages 224–233, 1998.

[MMPV11] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. Reasoning about strategies: On the model-checking problem. *CoRR*, abs/1112.6275, 2011.

[MMV10] Fabio Mogavero, Aniello Murano, and Moshe Y. Vardi. Reasoning about strategies. In *IARCS Annual Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 8, pages 133–144, 2010.

[MS87] ED Muller and PE Schupp. Alternating automata on infinite trees. *Theor. Comp. Sci.*, 54(2-3):267–276, October 1987.

[MS95] David E. Muller and Paul E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of rabin, mcnaughton and safra. *Theor. Comp. Sci.*, 141(1–2):69 – 107, 1995.

[Pin07] Sophie Pinchinat. Quantified mu-calculus with decision modalities for concurrent game structures. Technical report, Dept. of Computer Science, Faculty of Engineering and Information Technology, Australian National University, 2007.

[WvdHW07] Dirk Walther, Wiebe van der Hoek, and Michael Wooldridge. Alternating-time temporal logic with explicit strategies. In *Theoretical aspects of rationality and knowledge (TARK 07)*, pages 269–278. ACM, 2007.

Onera/Dtim, 2, avenue Édouard Belin, BP74025,, 31055 Toulouse Cedex 4, France
E-mail address: firstname.lastname@onera.fr

Onera/Dtim, 2, avenue Édouard Belin, BP74025,, 31055 Toulouse Cedex 4, France
E-mail address: firstname.lastname@onera.fr

Onera/Dtim, 2, avenue Édouard Belin, BP74025,, 31055 Toulouse Cedex 4, France
E-mail address: firstname.lastname@onera.fr