Updatable Strategy Logic

Christophe Chareton, Julien Brunel, David Chemouil

UPDATABLE STRATEGY LOGIC

RESEARCH REPORT

CHRISTOPHE CHARETON, JULIEN BRUNEL, AND DAVID CHEMOUIL

Abstract. In this article, we present Updatable Strategy Logic (USL), a multi-agent temporal logic 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 refine her strategy, that is to update it without revoking it. Syntactically, a new operator, called "unbinder", is introduced: it allows an agent to explicitly revoke a strategy, whereas revocation is implicit in SL. 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 temporal logics are receiving growing interest in contemporary research. Since the seminal work [2] of R. Alur, Th. A. Henzinger, and O. Kupferman on Alternating-Time Temporal Logic (ATL-ATL*), increasing efforts have been made to formalise agent interactions in game-theoretical terms.

Basically, multi-agent temporal logics enable to formulate assertions about the ability of agents to ensure temporal properties by following *strategies*. Thus, ATL-ATL* appears as a generalisation of CTL-CTL* in which the path quantifiers **E** and **A** are replaced by *strategy quantifiers*. They are interpreted in *Concurrent Game Structures* (CGSs), where agents make choices influencing the execution of the system.

In these logics, the ability to express strategy composition, which comes to nesting strategy quantifiers in a formula, is a major issue. Composition of strategies for different agents has notably been introduced in ATL$_{sc}$ [4, 9] and Strategy Logic (SL, presented first in [8] and then extended in [11, 12]. In the remainder of the article, we refer to the latter version of SL.). In these formalisms, the ability of agents to act upon the system is evaluated in contexts storing the behaviour of other agents. However, as noticed in [1], after a given agent has been bound to a strategy, if this very agent is bound to a new strategy, she *revokes* the former. Said otherwise, the new binding *overwrites* the previous one in the evaluation context.

This way, it is not possible to express what we call a *strategy refinement* for an agent *i.e.* the possibility to *enrich* her strategy. We envision at least two examples where we think this notion is relevant. In the remainder of the article, we will particularly focus on the second example.

Example 1 (Requirement modelling)**.** *The first application stems from our work in requirements modelling [5, 6]: let us consider two groups of agents G_1 and G_2, which share one agent a. In order to satisfy a property (or "requirement") p_1 (resp. p_2) every agent in G_1 (resp. G_2) follows a certain strategy. An interesting case occurs when G_2 is helped by G_1 to fulfill p_2. In other words, G_2 is able to ensure p_2 only in the context where G_1 follows its strategy for p_1. That is, we expect the strategies of agents in G_2 to compose with the strategies of agents in G_1. Since a belongs to both groups, her strategy for p_1 and her strategy for p_2 must be coherent so that there is a refinement of both strategies that satisfies both p_1 and p_2. Suppose now a third group of agents G_3 is added to the model, with requirement p_3. And suppose a is also in this group. Expressing that G_1 can*

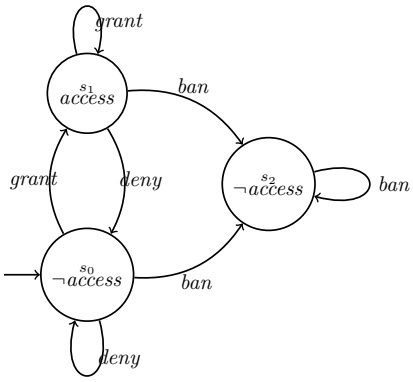

FIGURE 1. Model $\mathcal{M}_1$ for the server

follow a strategy which may be enriched either to help G_2 or to help G_3 imposes to consider two different potential improvement for that strategy. In this case, an explicit *formalism for strategy refinement is needed.*

This notion of refinement also allows to consider a particular kind of properties that we call *sustainable capability* (further explained in Sect. 3.1): a capability for an agent (*i.e.* an action the agent can perform and which constraints the set of possible next states) is sustainable if it remains usable 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* over it.

Example 2 (Sustainable capability)**.** *As an illustration (see Fig. 1), suppose a client program sends connection requests to a chat server. The server can always decide whether to* grant *or* deny access *to the client. In addition, the server can* ban *a client and refuse connection once and for all.*

We want to express with logical formulas (and to check them over the model) the fact the server is able to ensure some properties. For instance, the server is able to ban the client at any time, i.e. *to ensure that, from an arbitrary instant chosen by the server,* access *becomes false forever. As we will see more in details in Sect. 3.1 this property involves strategy revocation. We also want to express that the server is able to set* access *to true at any moment and as many times as it wants. This is the* sustainable capability *to ensure* access*. A stronger property is the ability to set* access *to true or to false at any moment and as many times as it wants. This is the* sustainable control *over* access*.*

In this paper, building on previous work introduced in [7], we present Updatable Strategy Logic (USL), a logic allowing to express these kinds of properties by considering strategies that are *updatable*, *i.e.* revocable or refinable. USL differs from SL in two main ways. On the one hand, the syntax of formulas features (1) a *binder* $(a \rhd x)$ enabling to associate a strategy to an agent *without overwriting strategies previously bound to her*; and (2) an *unbinder* $(a \not\rhd x)$ allowing an agent to revoke a strategy *explicitly.* On the other hand, the semantic framework is built to enable an agent to update her own strategy without revoking it.

The remainder of this paper is organized as follows: in Sect. 2, we define the syntax and semantics of USL, and we also describe the semantics for a restricted version of the logic featuring *memory-less strategies* only (we call the resulting logic USL^0). In Sect. 3, we detail the notions of sustainable capability and control, and we prove that these properties are not expressible in SL and that USL subsumes SL. Finally, we show in Sect. 4 that model-checking for USL is non-elementarily decidable and that it is PSPACE-complete for USL^0.

2. Syntax and Semantics

In this section we present the syntax and the semantics of USL, and a restricted semantics with memoryless strategies.

2.1. **Syntax.**

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

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle x \rangle\rangle \varphi \mid (A \rhd x)\varphi \mid (A \not\rhd x)\varphi \mid \varphi \mathbf{U} \varphi \mid \mathbf{X}\varphi$$

where $p \in At, A \subseteq Ag, x \in X$.

A set of agents is also called a *coalition*. The 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: $\Diamond\varphi = \top\mathbf{U}\varphi$ and $\Box\varphi = \neg\Diamond\neg\varphi$.

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 hereafter a notion of *free variables* $FV(\varphi)$ by induction, and then we call *sentence* is a formula φ s.t. $FV(\varphi) = \emptyset$.

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

2.2. **Semantics.**

USL interprets the refinement of a strategy as a precision given to this strategy. By refining her strategy, an agent will remove some non-determinism from it. The expression of strategy refinement thus relies on nondeterministic models.

Definition 2 (NATS)**.** *A Nondeterministic 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 $\mathcal{M}$, 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 s.t. two choices from different agents are always in non empty intersection: given a state $s \in M$ and agents a_1 and $a_2 \in Ag$, for every $c_1 \in Ch(a_1, s)$ and $c_2 \in Ch(a_2, s)$, $c_1 \cap c_2 \neq \emptyset$.*

The set of finite sequences of states that are possible in $\mathcal{M}$ is denoted by $Track : \tau = \tau_0\tau_1 \ldots \tau_k \in Track$ iff for every $i < k$, for every $a \in Ag$, there is $c_a \in \mathcal{P}(M)$ s.t. $c_a \in Ch(a, \tau_i)$ and $\tau_{i+1} \in c_a$. Similarly, an infinite sequence of states that are possible in $\mathcal{M}$, *i.e.* all its prefixes are in $Track$, 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 $i \in \mathbb{N}, \lambda^k_i = \lambda_{k+i}$.

Definition 3 (Strategies)**.** *A* strategy *σ is a map from $Ag \times Track$ to $\mathcal{P}(M)$ s.t. for every $(a, \tau), \sigma(a, \tau) \in Ch(a, last(\tau))$. We note $Strat$ the set of strategies in a model.*

Now we give the definition of contexts used for the interpretation of formulas. Unlike what is done for SL, we make a distinction between the part representing the active bindings (the commitments) and the part dealing with the strategy variables (the assignments).

Definition 4 (Assignments, commitments and contexts)**.** *An* assignment α *is a partial function from* X *to* $Strat$ *which associates a strategy variable with to a strategy in the model.*

A commitment κ *is a finite sequence upon* $\mathcal{P}(Ag) \times X$*, which stores the bindings of sets of agents (or coalitions) to strategy variables. Let* κ_1 *and* κ_2 *be two commitments, then we note* $\kappa_1 \cdot \kappa_2$ *their concatenation.*

A context χ *is a pair of an assignment and a commitment.*

A context induces a function from $Track$ to $\mathcal{P}(M)$. This function, which we denote the same way as the context itself, yields a set of possible successor states, following the execution of a track. These are defined according to the strategies that are present in the context and which only concern the agents bound to some strategy variable via a commitment.

Definition 5 (Context function)**.** *Let* $\chi = (\alpha, \kappa)$ *be a context and write* $\kappa_\emptyset$ *for the empty sequence upon* $\mathcal{P}(Ag) \times X$*. The function induced by* χ *is defined as follows:*

- $(\alpha, \kappa_\emptyset)(\tau) = M$
- $(\alpha, (A, x))(\tau) =$
  - $\bigcap_{a \in A} \alpha(x)(a, \tau)$ *if* $A \neq \emptyset$
  - M *otherwise*
- $(\alpha, \kappa \cdot (A, x))(\tau) =$
  - $(\alpha, \kappa)(\tau) \cap (\alpha, (A, x))(\tau)$ *if this intersection is not empty*
  - $(\alpha, \kappa)(\tau)$ *otherwise (which means that in case the commitment* (A, x) *is in contradiction with* κ*, we do not take it into account and we keep* κ*).*

Definition 6 (Outcome)**.** *Let* χ *be a context and* s *be a state. The* outcome *of* χ *and* s*, written* $out(\chi, s)$*, is defined as the set of paths derived by* χ *from* s*: let* λ *be a path, then* $\lambda \in out(\chi, s)$ *iff* $s = \lambda_0$ *and for every* $n \in \mathbb{N}$*,* $\lambda_{n+1} \in \chi(\lambda_0 \dots \lambda_n)$*.*

Definition 7 (Strategy and assignment translation)**.** *Let* σ *be a strategy and* τ *be a track. Then* σ^τ *is the strategy s.t. for every* $\tau' \in Track$*,* $\sigma^\tau(\tau') = \sigma(\tau \cdot \tau')$*. The notion is extended to an assignment* $\alpha : \alpha^\tau$ *is the assignment with same domain as* α *and s.t. for every* $x \in dom(\alpha), \alpha^\tau(x) = (\alpha(x))^\tau$*.*

Definition 8 (Transformations of contexts)**.** *Given a commitment* κ*, coalitions* A *and* B*, a strategy variable* x*, an assignment* α *and a strategy* σ*, we define transformations* $[A \to x], [A \nrightarrow x]$ *and* $[x \to \sigma]$ *as follows:*

Binding: $\kappa[A \to x] = \kappa \cdot (A, x)$
Revocation: $\kappa_\emptyset[A \nrightarrow x] = \kappa_\emptyset$ *and* $((B, x) \cdot \kappa)[A \nrightarrow x] = (B \backslash A, x) \cdot (\kappa[A \nrightarrow x])$
Strategy instantiation: $\alpha[x \to \sigma]$ *is the assignment with domain* $dom(\alpha) \cup \{x\}$ *s.t.* $\forall y \in dom(\alpha) \backslash \{x\}, \alpha[x \to \sigma](y) = \alpha(y)$ *and* $\alpha[x \to \sigma](x) = \sigma$*.*

Definition 9 (Satisfaction relation)**.** *Let* $\mathcal{M}$ *be a NATS, then for every assignment* α*, commitment* κ *and path* λ*:*

- $\mathcal{M}, \alpha, \kappa, \lambda \models p$ *iff* $p \in v(\lambda_0)$*, with* $p \in At$
- $\mathcal{M}, \alpha, \kappa, \lambda \models \neg\varphi$ *iff it is not true that* $\mathcal{M}, \alpha, \kappa, \lambda \models \varphi$
- $\mathcal{M}, \alpha, \kappa, \lambda \models \varphi_1 \wedge \varphi_2$ *iff* $\mathcal{M}, \alpha, \kappa, \lambda \models \varphi_1$ *and* $\mathcal{M}, \alpha, \kappa, \lambda \models \varphi_2$
- $\mathcal{M}, \alpha, \kappa, \lambda \models \langle\langle x \rangle\rangle \varphi$ *iff there is a strategy* $\sigma \in Strat$ *s.t.* $\mathcal{M}, \alpha[x \to \sigma], \kappa, \lambda \models \varphi$
- $\mathcal{M}, \alpha, \kappa, \lambda \models (A \rhd x)\varphi$ *iff for every* λ' *in* $out((\alpha, \kappa[A \to x]), \lambda_0), \mathcal{M}, \alpha, \kappa[A \to x], \lambda' \models \varphi$
- $\mathcal{M}, \alpha, \kappa, \lambda \models (A \not\rhd x)\varphi$ *iff for all* λ' *in* $out((\alpha, \kappa[A \nrightarrow x]), \lambda_0), \mathcal{M}, \alpha, \kappa[A \nrightarrow x], \lambda' \models \varphi$
- $\mathcal{M}, \alpha, \kappa, \lambda \models \mathbf{X}\psi$ *iff* $\mathcal{M}, \alpha^{\lambda_0}, \kappa, \lambda^1 \models \psi$
- $\mathcal{M}, \alpha, \kappa, \lambda \models \psi_1 \mathbf{U} \psi_2$ *iff there is* $i \in \mathbb{N}$ *s.t.* $\mathcal{M}, \alpha^{\lambda_0 \dots \lambda_{i-1}}, \kappa, \lambda^i \models \psi_2$ *and for every* $0 \leq j < i, \mathcal{M}, \alpha^{\lambda_0 \dots \lambda_{j-1}}, \kappa, \lambda^j \models \psi_1$

We write $\mathcal{M}, \alpha, \kappa, s \models \varphi$ if for every path λ s.t. $\lambda_0 = s$, we have $\mathcal{M}, \alpha, \kappa, \lambda \models \varphi$. Furthermore, let $\alpha_\emptyset$ be the unique assignment with empty domain. Then we write $\mathcal{M}, s \models \varphi$ iff $\mathcal{M}, \alpha_\emptyset, \kappa_\emptyset, s \models \varphi$.

Let us comment these definitions: for every context $\chi = (\alpha, \kappa)$ and state s, the definition of $out(\chi, s)$ ensures that the different binders encoded in χ compose their choices together, *as far as possible*. In case two contradictory choices from an agent are encoded in the context, the priority is given to the first binding that was introduced in this context (the *leftmost binding* in the formula). This guarantees that a formula requiring the composition of two contradictory strategies is false. For instance, suppose that $\langle\!\langle x_1 \rangle\!\rangle (a \rhd x_1)\varphi_1$ and $\langle\!\langle x_2 \rangle\!\rangle (a \rhd x_2)\varphi_2$ 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, $\langle\!\langle x_1 \rangle\!\rangle (a \rhd x_1)(\varphi_1 \wedge \langle\!\langle x_2 \rangle\!\rangle (a \rhd x_2)\varphi_2)$ 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.

2.3. **USL^0.** USL^0 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 modelled by using memory-less strategies only.

Definition 10 (Memory-less strategies)**.** *A memory-less strategy is a function σ from $Ag \times M$ to $\mathcal{P}(M)$ s.t. for every $(a, s) \in Ag \times M, \sigma(a, s) \in Ch(a, s)$. We note $Strat^0$ the set of memory-less strategies in a model.*

All the semantic definitions for USL adapt to USL^0 by simply modifying the clause for $\langle\!\langle x \rangle\!\rangle$: the strategy variable ranges over $Strat^0$ instead of $Strat$ in the case of USL^0.

3. Sustainability, USL and SL

In this section we further analyse the mechanisms of strategy update. This analysis is led by help of sustainable capability and sustainable control and by comparison with SL. First, we give an informal explanation of these concepts and exhibit their expression in USL. Then we prove that they cannot be expressed within SL. In the remainder, the notations used for SL formulas and semantic framework are the ones defined in [11].

3.1. **Sustainable Capability and Sustainable Control in USL.** Here we introduce the notions of sustainable capability and sustainable control, by help of Example 2.

3.1.1. *A Capability that is Not Sustainable.* First, let us consider the property "*the server* (*S*) can always ensure *access* in the next state". In SL, if *S* is the only agent, the property is expressed by Formula 1:

$$\langle\!\langle x_1 \rangle\!\rangle (S, x_1) \Box (\langle\!\langle x_2 \rangle\!\rangle (S, x_2) \mathbf{X} access) \tag{1}$$

As proved in Sect. 3.2, it is equivalent to the USL formula:

$$\langle\!\langle x_1 \rangle\!\rangle (S \rhd x_1) \Box (\langle\!\langle x_2 \rangle\!\rangle (S \not\rhd x_1)(a \rhd x_2) \mathbf{X} access) \tag{2}$$

Now and in the remaining of this paper, for any variable x, we write σ_x a strategy instantiating x. In Formula 1, the subformula $\langle\!\langle x_2 \rangle\!\rangle (S, x_2) \mathbf{X} access$ states that S can adopt a strategy σ_{x_2} that ensures $\mathbf{X} access$, even if σ_{x_2} is in contradiction with σ_{x_1} and makes it lose its ability to ensure $\mathbf{X} access$ later on.

Consider the model $\mathcal{M}_2$ illustrated by Fig. 2. 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 S has a choice c to go to s' from s. From state s_0, S can either choose to stay in that state (choice *deny*) , where *access* does not hold, or to go to state s_1 (choice *grant*) where *access* holds. In the latter case, the server will not be able to leave s_2, in which the connection is closed.

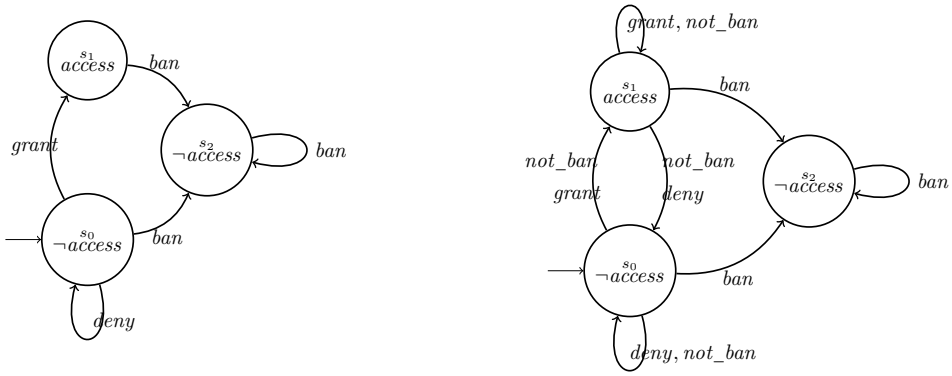

FIGURE 2. Models $\mathcal{M}_2$ (left) and $\mathcal{M}_3$ (right) for the server

Formula 1 holds in state s_0 of $\mathcal{M}_2$. Indeed, by playing strategy *always-play-deny* (apd, defined by: for all $\tau \in Track$, $apd(a, \tau) = deny$), S remains in s_0, from which it can ensure $\mathbf{X}access$. However, in order to ensure $access$ once, S must revoke apd and adopt a new strategy with choice $grant$. Doing this, S loses its capability to ensure $\mathbf{X}access$. It can achieve $\mathbf{X}access$ at any time, but only once: its capability is not sustainable. This somehow contradicts the intuition of the property we want to express. By "*S* can always ensure *access* in the next state", we mean that *S* remains able to ensure *access* even after using this capability.

3.1.2. *Sustainable Capability.* Consider now the following USL formula that is syntactically similar to Formula 1:

$$\langle\langle x_1 \rangle\rangle (S \rhd x_1) \Box (\langle\langle x_2 \rangle\rangle (S \rhd x_2) \mathbf{X} access) \tag{3}$$

USL requires that the strategies σ_{x_1} and σ_{x_2}, instantiating x_1 and x_2, can be composed together. So the strategy σ_{x_2} does not revoke σ_{x_1} according to which S has the capability to ensure $\mathbf{X}access$ later on in the execution. Thus, Formula 3 expresses that S is able to ensure $\mathbf{X}access$ as many times as it wants. This property is not satisfied in $\mathcal{M}_2$.

Let us now enrich the model $\mathcal{M}_1$ from Fig. 1 with the nondeterministic choice not_ban (see model $\mathcal{M}_3$ in Fig. 2). This choice unifies the choices $grant$ and $deny$, it corresponds to *not banning* the client from the service. (Now we consider non-deterministic choices, in the figure, a label c from s to s' means that "from s, S has a choice c that *includes* s'".)

As formally stated in Example 3 hereunder, Formula 3 is true in state s_0 of $\mathcal{M}_3$ with the strategy *always-play-not_ban* (apn, defined by: for all $\tau \in Track$, $apn(S, \tau) = not\_ban$) instantiating x_1. Indeed, this strategy can be refined by making the additional choice $grant$ (which ensures $\mathbf{X}access$) at any time. With this strategy, S always remains capable of ensuring $\mathbf{X}access$, as many times as it wants: it has sustainable capability to ensure $\mathbf{X}access$.

Example 3 (Proposition). $\mathcal{M}_3, s_0 \models \langle\langle x_1 \rangle\rangle (S \rhd x_1) \Box (\langle\langle x_2 \rangle\rangle (S \rhd x_2) \mathbf{X} access)$.

Proof. Let us prove the proposition of Example 3 formally. First, $out((\langle (x_1 \mapsto apn) \rangle, (S, x_1)), s_0) = s_0 \cdot (s_0 + s_1)^\omega$, where $(s_0 + s_1)^\omega$ is the notation taken from language theory for the set of infinite sequences of states s_0 or s_1 (we also write $(s_0 + s_1)^*$ the set of such finite sequences). Thus, for all

$\lambda \in out((\langle\langle(x_1 \mapsto apn)\rangle\rangle, (S, x_1)), s_0)$, for all $n \in \mathbb{N}, \lambda_n \in \{s_0, s_1\}, out((\langle\langle(x_1 \mapsto apn)\rangle\rangle^{\lambda_0 \dots \lambda_{n-1}}, (S, x_1)), \lambda_n) = \lambda_n \cdot (s_0 + s_1)^\omega$.

Now, consider the strategy *play-grant-once*(pgo) defined over $S \times (s_0 + s_1)^*$ by, for $i \in \{0, 1\}$: $pgo(S, s_i) = grant$ and $pgo(S, s_i \cdot \tau) = not\_ban$ for all $\tau \in (s_0 + s_1)^*$.

It is easy to see that for all $\lambda' \in out((\langle\langle(x_1 \mapsto apn^{\lambda_0 \dots \lambda_{n-1}}), (x_2 \mapsto pgo)\rangle\rangle, ((S, x_1), (S, x_2))), \lambda_n)$, we have $\lambda'_1 = s_1$.

Now we can derive the steps of evaluation: for all $\lambda \in out((\langle\langle(x_1 \mapsto apn)\rangle\rangle, (S, x_1)), s_0)$, for all $n \in \mathbb{N}$, for all $\lambda' \in out((\langle\langle(x_1 \mapsto apn)^{\lambda_0 \dots \lambda_{n-1}}, (x_2 \mapsto pgo)\rangle\rangle, ((S, x_1), (S, x_2))), \lambda_n)$:

$$\begin{array}{l}
\mathcal{M}_3, (\langle\langle(x_1 \mapsto apn)^{\lambda_0 \dots \lambda_n}, (x_2 \mapsto pgo^{\lambda_n})\rangle\rangle, ((S, x_1), (S, x_2)), \lambda'^1 \models access \\
\mathcal{M}_3, \langle\langle(x_1 \mapsto apn)^{\lambda_0 \dots \lambda_{n-1}}, (x_2 \mapsto pgo)\rangle\rangle, ((S, x_1), (S, x_2)), \lambda'_0 \models \mathbf{X} access \\
\mathcal{M}_3, \langle\langle(x_1 \mapsto apn)^{\lambda_0 \dots \lambda_{n-1}}, (x_2 \mapsto pgo)\rangle\rangle, (S, x_1), \lambda_{n-1} \models \mathbf{X}(S \rhd x_2) access \\
\mathcal{M}_3, (\langle\langle(x_1 \mapsto apn^{\lambda_0 \dots \lambda_{n-1}})\rangle\rangle, (S, x_1), \lambda_{n-1} \models \langle\langle x_2 \rangle\rangle (S \rhd x_2) \mathbf{X} access \\
\mathcal{M}_3, \langle\langle(x_1 \mapsto apn)\rangle\rangle, (S, x_1), s_0 \models \Box(\langle\langle x_2 \rangle\rangle (S \rhd x_2) \mathbf{X} access) \\
\mathcal{M}_3, \langle\langle(x_1 \mapsto apn)\rangle\rangle, \kappa_\emptyset, s_0 \models (S \rhd x_1) \Box(\langle\langle x_2 \rangle\rangle (S \rhd x_2) \mathbf{X} access) \\
\mathcal{M}_3, s_0 \models \langle\langle x_1 \rangle\rangle (S \rhd x_1) \Box(\langle\langle x_2 \rangle\rangle (S \rhd x_2) \mathbf{X} access)
\end{array}$$

□

3.1.3. *Sustainable Control.* But S is not only sustainably capable of ensuring $access$. A similar strategy *play-deny-once* can also be used at any time to refine strategy *apn* so as to ensure $\mathbf{X}\neg access$. So, S is in fact sustainably capable of deciding whether or not *access* holds. We say that, in $\mathcal{M}_3$, S has *sustainable control* over the property *access* from state s. Formally:

$$\mathcal{M}_3, s_0 \models \langle\langle x \rangle\rangle (S \rhd x) \Box(\langle\langle y \rangle\rangle (S \rhd y) \mathbf{X} access \wedge \langle\langle y \rangle\rangle (S \rhd y) \mathbf{X} \neg access) \tag{4}$$

3.1.4. *Strategy Revocation.* Actually, strategy *apn* is also *always revocable*. When following it, S can revoke it at any time to ban the client. This is shown in Formula 5 which ends up the description of the server capabilities, as presented in Example 2 in Sect. 1:

$$\begin{array}{l}
\mathcal{M}_3, s_0 \models \langle\langle x \rangle\rangle (S \rhd x) \Box(\langle\langle y \rangle\rangle (S \rhd y) \mathbf{X} access \\
\qquad\qquad \wedge \langle\langle y \rangle\rangle (S \rhd y) \mathbf{X} \neg access \wedge \langle\langle z \rangle\rangle (S \not\rhd x)(S \rhd z) \Box \neg access)
\end{array} \tag{5}$$

3.2. **Comparison with 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: 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). Still, we show that sustainable control cannot be satisfied in the class of deterministic models. Furthermore, even with a natural extension of SL semantics to nondeterministic models, sustainable control is not expressible in SL. First, let us give a formal definition of sustainable control, generalising the property expressed in Formula 4.

Definition 11 (Sustainable control)**.** *Given a coalition of agents A and $\varphi \in$ USL, the sustainable control of A over φ is the class of models $\mathcal{M}$ with a state s in its domain such that:*

$$\mathcal{M}, s \models \langle\langle x_1 \rangle\rangle (A \rhd x_1) \Box(\langle\langle x_2 \rangle\rangle (A \rhd x_2) \mathbf{X} \varphi \wedge \langle\langle x_2 \rangle\rangle (A \rhd x_2) \mathbf{X} \neg \varphi) \tag{6}$$

The main results of this section are stated as follows:

- There is an embedding of SL models and formulas into USL models and formulas (resp.). It preserves the satisfaction relation.

- Sustainable control has no deterministic instance. Therefore one needs the nondeterminism of NATSs to express it.
- Sustainable control over an atomic proposition is not expressible under a natural extension of SL semantics to NATSs. Strategy refinement is also needed to express it.

3.2.1. *Embedding of SL into USL.* The embedding of SL into USL consists in a parallel transformation from SL models and formulas to USL models (Concurrent Games Structures, or CGSs) and formulas. This transformation preserves the satisfaction relation.

Let us first bring the necessary definitions related to SL:
SL. This paragraph only gives a brief presentation of SL syntax and semantics. The non familiar reader may refer to [11] for a complete presentation of this framework.

Definition 12 (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 $a \in Ag$ and $x \in X$.

SL uses a notion of *free agents* of a formula φ ($Fa(\varphi)$), defined inductively as follows:

- $Fa(p) = \emptyset$, for $p \in At$
- $Fa(\neg\varphi) = Fa(\langle\langle x\rangle\rangle\varphi) = Fa([\![x]\!]\varphi) = Fa(\varphi)$
- $Fa(\varphi_1 \wedge \varphi_2) = Fa(\varphi_1 \vee \varphi_2)Fa(\varphi_1) \cup Fa(\varphi_2)$
- $Fa(\varphi_1\mathbf{U}\varphi_2) = Fa(\varphi_1\mathbf{R}\varphi_2) = Fa(\mathbf{X}\varphi) = Ag$
- $Fa((a, x \rhd \varphi)) = Fa(\varphi)\backslash\{a\}$

The sentences of SL are the formulas φ such that $FV(\varphi) = Fa(\varphi) = \emptyset$, where $FV(\varphi)$ is defined similarly as for USL.

Here is the definition of CGSs:

Definition 13. *A CGS is a tuple $\mathcal{G} = \langle Ag, M, At, v, Ac, Tr\rangle$ where :*

- *Ag, M, At and v are as in NATSs.*
- *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 14. *Let $\mathcal{G}$ be a CGS, the set $Path$ is the set of infinite sequences $\lambda_0\lambda_1\ldots$ s.t. for every $n \in \mathbb{N}$, there is $Dc \in DC$ s.t. $Tr(\lambda_n, Dc) = \lambda_{n+1}$. $Track$ is the set of such finite sequences. A strategy in CGS $\mathcal{G}$ is a partial function from $Track$ to the set of actions. We note $Strat$ the set of strategies in a model:*

$$Strat = \{\sigma : Track \to Ac\}$$

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 $(\mathcal{G}, \chi, s)$ where:

- $\mathcal{G}$ is a CGS.
- χ 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 $(\chi, s)^n$ the pair defined by:

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

where the notation χ^τ, for χ a strategy context and τ a track, is as in Def. 7. Again, the definition for the semantics of SL proceeds in two steps:

Definition 15 (Satisfaction relation for SL)**.** *Let $\mathcal{G}$ be a CGS, χ a strategy context for $\mathcal{G}$ and s a state in it. Then:*

(1) $\mathcal{G}, \chi, s \models_{SL} p$ iff $p \in \lambda(s)$, with $p \in At$.
(2) For every formula φ, φ_1 and φ_2 :
  (a) $\mathcal{G}, \chi, s \models_{SL} \neg\varphi$ iff it is not true that $\mathcal{G}, \chi, s \models_{SL} \varphi$.
  (b) $\mathcal{G}, \chi, s \models_{SL} \varphi_1 \wedge \varphi_2$ iff $\mathcal{G}, \chi, s \models_{SL} \varphi_1$ and $\mathcal{G}, \chi, s \models_{SL} \varphi_2$.
  (c) $\mathcal{G}, \chi, s \models_{SL} \varphi_1 \vee \varphi_2$ iff $\mathcal{G}, \chi, s \models_{SL} \varphi_1$ or $\mathcal{G}, \chi, s \models_{SL} \varphi_2$.
(3) For every variable x and formula φ :
  (a) $\mathcal{G}, \chi, s \models_{SL} \langle\langle x \rangle\rangle \varphi$ iff there is an s-total strategy f s.t. $\mathcal{G}, \chi[x \to f], s \models_{SL} \varphi$.
  (b) $\mathcal{G}, \chi, s \models_{SL} [\![x]\!] \varphi$ iff for every s-total strategy f, $\mathcal{G}, \chi[x \to f], s \models_{SL} \varphi$.
(4) For every agent a, variable x and formula φ, $\mathcal{G}, \chi, s \models_{SL} (a, x)\varphi$ iff $\mathcal{G}, \chi[\chi(x) \backslash \chi(a)], s \models_{SL} \varphi$.
(5) If χ is a complete strategy context, then for every formula φ, φ_1 and φ_2 :
  (a) $\mathcal{G}, \chi, s \models_{SL} \mathbf{X}\varphi$ iff $\mathcal{G}, (\chi, s)^1 \models_{SL} \varphi$.
  (b) $\mathcal{G}, \chi, s \models_{SL} \varphi_1 \mathbf{U} \varphi_2$ iff there is $i \in \mathbb{N}$ s.t. $\mathcal{G}, (\chi, s)^i \models_{SL} \varphi_2$ and, for every index $j \in \mathbb{N}$ s.t. $0 \leq i < j$, $\mathcal{G}, (\chi, s)^j \models_{SL} \varphi_1$.
  (c) $\mathcal{G}, \chi, s \models_{SL} \varphi_1 \mathbf{R} \varphi_2$ iff for every index $i \in \mathbb{N}$, it holds that $\mathcal{G}, (\chi, s)^i \models_{SL} \varphi_2$ or there is an index $j \in \mathbb{N}$ s.t. $0 \leq j \leq i$, $\mathcal{G}, (\chi, s)^j \models_{SL} \varphi_1$

Where $\chi[\chi(x) \backslash \chi(a)]$ is obtained from χ by giving for a the value $\chi(x)$.

Definition 16 (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 $\mathcal{G}, \chi_\emptyset, s \models_{SL} \varphi$, where $\chi_\emptyset$ is the context with empty domain.

Now that the required definitions are given, we can come to the embedding of SL into USL:
Translation of SL into USL.

Proposition 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 $\overline{\varphi}$ in USL s.t. for every CGS $\mathcal{G}$ and for every $\varphi \in SL$, $\mathcal{G} \models \varphi$ iff $\mathcal{M}_\mathcal{G} \models \overline{\varphi}$. Furthermore, if we consider a single agent in the language, the transformation of models reduces to the interpretation of actions as choices.*

In SL, choices from USL are replaced by actions, and agents playing along the same strategies perform the same action at every state. To prove Prop. 1, first we define both the syntactical and semantic transformations that enable to express, by help of new atomic symbols, the related 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 $Dc, Dc' \in DC$, $Dc \equiv Dc'$ iff $\forall a, b \in Ag, (Dc(a) = Dc(b))$ iff $(Dc'(a) = Dc'(b))$. 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 $Dc' \in DC$, $Tr'((s, Dc), Dc') = (Tr(s, Dc'), Dc')$.

Then we have the following lemma:

Lemma 1.

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

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-tuple (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): $\mathcal{G}', \alpha, \kappa, s \models_{\text{NATS}} (a, x)\psi$ iff $\mathcal{G}', \alpha, \kappa[\kappa(x)\backslash\kappa(a)], s \models (a, x)\psi$. Since we are using it only in the model $\mathcal{G}'$ with deterministic choices, $\models_{\text{NATS}}$ does not need further definition so far. An extension of $\models_{\text{NATS}}$ for every NATSs is given further. 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.

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 $[A \rhd x]$ for USL by: for any formula φ, coalition A and variable x, $[A \rhd x]\varphi \triangleq (A \not\triangleright X)(A \rhd x)\varphi$. Then $[A \rhd x]$ 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 $(Ag \not\rhd X)$ $(A \rhd x)$, an ATL-like capability operator is also definable for USL).

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

Lemma 2.

$$\mathcal{G}' \models_{NATS} \varphi' \textit{ iff } \mathcal{G}' \models \overline{\varphi}$$

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

The two preceding lemmas bring Prop. 1.

3.2.2. *Sustainable Control has no Deterministic Instance.* Let us call DNATS a NATS with deterministic choices: *i.e.* a NATS $\mathcal{M} = \langle Ag, M, At, v, Ch\rangle$ is a DNATS iff for every $s \in M$ and for every function C from Ag to $\mathcal{P}(M)$ such that for every $a \in Ag, C(a) \in Ch(a,s), \bigcap_{a\in Ag} C(a)$ is a singleton. The following proposition states that sustainable control has no deterministic instance.

Proposition 2. *For every DNATS $\mathcal{M}$, for every $s \in M, a \in Ag, \varphi \in$ USL:*

$$\mathcal{M}, s \not\models \langle\!\langle x\rangle\!\rangle(a \rhd x)\Box(\langle\!\langle y\rangle\!\rangle(a \rhd y)\mathbf{X}\varphi \wedge \langle\!\langle y\rangle\!\rangle(a \rhd y)\mathbf{X}\neg\varphi)$$

Proof. Suppose that there is a DNATS $\mathcal{M}$ with set of agents Ag, and a state s s.t. $\mathcal{M}, s \models \langle\!\langle x\rangle\!\rangle(a \rhd x)\Box(\langle\!\langle y\rangle\!\rangle(a \rhd y)\mathbf{X}\varphi \wedge \langle\!\langle y\rangle\!\rangle(a \rhd y)\mathbf{X}\neg\varphi)$. Let us call $Ag\backslash\{a\}-$complete a context $\chi = (\alpha, \kappa)$ such that every $a' \in Ag\backslash\{a\}$ appears in κ and is committed to a variable strategy distinct from x and y and in the domain of α. Now, $\mathcal{M}, s \models \langle\!\langle x\rangle\!\rangle(a \rhd x)\Box(\langle\!\langle y\rangle\!\rangle(a \rhd y)\mathbf{X}\varphi \wedge \langle\!\langle y\rangle\!\rangle(a \rhd y)\mathbf{X}\neg\varphi)$ iff there is a strategy σ_x s.t. $\mathcal{M}, \langle(x \mapsto \sigma_x)\rangle, (a, x), s \models \Box(\langle\!\langle y\rangle\!\rangle(a \rhd y)\mathbf{X}\varphi \wedge \langle\!\langle y\rangle\!\rangle(a \rhd y)\mathbf{X}\neg\varphi)$. One checks that this is iff there is a strategy σ_x s.t. for all $Ag\backslash\{a\}-$complete context $\chi = (\alpha, \kappa), \mathcal{M}, \alpha[x \mapsto \sigma_x], \kappa[a \to x], s \models \Box(\langle\!\langle y\rangle\!\rangle(a \rhd y)\mathbf{X}\varphi \wedge \langle\!\langle y\rangle\!\rangle(a \rhd y)\mathbf{X}\neg\varphi)$ Because $\mathcal{M}$ is a DNATS, for all $Ag\backslash\{a\}-$complete context, $out((\alpha[x \mapsto \sigma_x], \kappa[a \to x]), s)$ is a unique path $\lambda = \lambda_0\lambda_1 \ldots$ in $\mathcal{M}$.

Now, next steps in the evaluation of $\mathcal{M}, s \models \langle\!\langle x\rangle\!\rangle(a \rhd x)\Box(\langle\!\langle y\rangle\!\rangle(a \rhd y)\mathbf{X}\varphi \wedge \langle\!\langle y\rangle\!\rangle(a \rhd y)\mathbf{X}\neg\varphi)$ require, for every $i \in \mathbb{N}$, that $\mathcal{M}, \langle\alpha[x \mapsto \sigma_x]\rangle^{\lambda_0\ldots\lambda_{i-1}}, \kappa[a \to x], \lambda^i \models \langle\!\langle y\rangle\!\rangle(a \rhd y)\mathbf{X}\varphi$ and $\mathcal{M}, \langle\alpha[x \mapsto \sigma_x]\rangle^{\lambda_0\ldots\lambda_{i-1}}, \kappa[a \to x], \lambda^i \models \langle\!\langle y\rangle\!\rangle(a \rhd y)\mathbf{X}\neg\varphi$. And the following steps require for every $i \in \mathbb{N}$, that there are strategies σ_{y_1} and σ_{y_2} s.t. for every $\lambda' \in out(\langle\langle\alpha[x \mapsto \sigma_x]\rangle^{\lambda_0\ldots\lambda_{i-1}}\rangle[y \to \sigma_{y_1}^{\lambda_i}]\rangle, \kappa[a \to x][a \to y_1], \lambda_i), \mathcal{M}, (\langle\langle\alpha[x \mapsto \sigma_x]\rangle^{\lambda_0\ldots\lambda_{i-1}}\rangle[y \to \sigma_{y_1}^{\lambda_i}]\rangle, \kappa[a \to x][a \to y_1]), \lambda'^1 \models \varphi$ and for every $\lambda' \in out(\langle\langle\alpha[x \mapsto \sigma_x]\rangle^{\lambda_0\ldots\lambda_{i-1}}\rangle[y \to \sigma_{y_2}^{\lambda_i}]\rangle, \kappa[a \to x][a \to y_2], \lambda_i), \mathcal{M}, (\langle\langle\alpha[x \mapsto \sigma_x]\rangle^{\lambda_0\ldots\lambda_{i-1}}\rangle[y \to \sigma_{y_2}^{\lambda_i}]\rangle, \kappa[a \to x][a \to y_2]), \lambda'^1 \models \neg\varphi$.

But for every $k \in \{1, 2\}$, $out(\langle\langle\alpha[x \mapsto \sigma_x]\rangle^{\lambda_0\ldots\lambda_{i-1}}\rangle[y \to \sigma_{y_i}^{\lambda_i}]\rangle, \kappa[a \to x][a \to y_i], \lambda_i)$ 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 $i \in \mathbb{N}, \mathcal{M}, \langle\alpha[x \mapsto \sigma_x]\rangle^{\lambda_0\ldots\lambda_{i-1}}, \kappa[a \to x], \lambda_{i+1} \models \varphi$ and $\mathcal{M}, \langle\alpha[x \mapsto \sigma_x]\rangle^{\lambda_0\ldots\lambda_{i-1}}, \kappa[a \to x], \lambda_{i+1} \models \varphi$, which gives a contradiction.

□

Remark 1. *From Proposition 2 we can easily deduce that the models of SL (CGSs), which are deterministic, do not allow to express sustainable control. But since sustainable control is formally defined in term of NATSs, the fact CGSs cannot express it would be tricky to establish. However, in the case of one single agent, the mapping from CGSs to NATSs (defined in the embedding from SL[1] to USL) is identity (more precisely it reduces to the interpretation of actions in CGSs as choices in NATSs). Then we easily see that CGSs are "equivalent" to DNATSs, and thus do not allow to express sustainable control.*

3.2.3. *Sustainable Control over an Atomic Proposition is not Expressible in SL over NATSs.* The following argument uses an extension of SL semantics to nondeterministic NATSs. We already have a straightforward interpretation of SL over DNATSs. The generalisation to NATSs requires to define an interpretation of nondeterministic choices for agents. Indeed, the difference between DNATSs and NATSs lies in the fact that under DNATSs, a context χ defining a strategy for each agent induces an unique possible path λ. The definition of SL sentences induces that their temporal operator can only be evaluated in such contexts. So, the evaluation of temporal operators follows the classical definition for semantics of LTL under paths. Under NATSs, χ induces a non-empty set of paths with possibly more than one element. Our generalisation follows the semantics of LTL under Kripke models, taking the universal quantification over paths in $out(\chi, s)$.

Definition 17 (Semantics for SL under NATSs)**.** *Let $\mathcal{M}$ be a NATS for a language with set of variables X, λ a path, (α, κ) a context for $\mathcal{M}$ and φ, φ_1 and φ_2 formulas in SL, then:*

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

Notice that in the interpretation of (a, x), agent a revokes her current strategy and is then bound to the strategy x.

Proposition 3. *Sustainable control over an atomic proposition is not expressible in SL over NATSs.*

To prove this proposition. First we prove a similar result for SL^1 (Proposition 4). Then we prove it generalises to SL.

Proposition 4. *Sustainable control over an atomic proposition is not expressible in SL^1 over NATSs.*

To prove it, first we consider the set of SL^1 formulas $\{\theta_i\}_{i \in \mathbb{N}}$, each one asserting that a is indefinitely capable of deciding 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:

- $\theta_0 := \langle\langle x \rangle\rangle(a, x)\Box(\langle\langle x \rangle\rangle(a, x)\mathbf{X}p \wedge \langle\langle x \rangle\rangle(a, x)\mathbf{X}\neg p)$
- $\theta_1 := \langle\langle x \rangle\rangle(a, x)\Box(\langle\langle x \rangle\rangle(a, x)\mathbf{X}(p \wedge \Box(\langle\langle x \rangle\rangle(a, x)\mathbf{X}p \wedge \langle\langle x \rangle\rangle(a, x)\mathbf{X}\neg p))$
  $\wedge \langle\langle x \rangle\rangle(a, x)\mathbf{X}(\neg p \wedge \Box(\langle\langle x \rangle\rangle(a, x)\mathbf{X}p \wedge \langle\langle x \rangle\rangle(a, x)\mathbf{X}\neg p)))$
- 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 use the following notation $\theta_\infty := \langle\langle x\rangle\rangle(a \rhd x)\Box(\langle\langle y\rangle\rangle(a \rhd y)\mathbf{X}p \wedge \langle\langle y\rangle\rangle(a \rhd y)\mathbf{X}\neg p)$. θ_∞ under USL is equivalent to $\{\theta_i\}_{i\in\mathbb{N}}$. Let us suppose that there is a formula $\Theta \in \mathrm{SL}^1$ equivalent to θ_∞ 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 $\Box$.

The proof of Prop. 4 proceeds by an *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 $\mathrm{SL}^1_{\mathrm{R}}$.
Θ is in $\mathrm{SL}^1_{\mathrm{R}}$.

Lemma 3. *Θ, if it exists, is in Σ_1-SL*

Proof. 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 using p as only atom, $\langle\langle x\rangle\rangle(a, x)\varphi$ 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 4. *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 $\bigvee_{0\leq k\leq|\Theta|}(\mathbf{X}^k\psi_2 \wedge \bigwedge_{0\leq i<k}\mathbf{X}^i\psi_1)$, where $\mathbf{X}^k$ stands for a sequence of $\mathbf{X}$ of length k. In this transformation, the disjunction $\bigvee_{0\leq k\leq|\Theta|}$ 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 $\varphi_1\mathbf{R}\varphi_2 \leftrightarrow (\varphi_2\mathbf{U}(\varphi_1 \wedge \varphi_2)) \vee \Box\varphi_1$. □

Θ' is not in $\mathrm{SL}^1_{\mathrm{R}}$. 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* (we call *at-most-binary-trees* trees s.t. each 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 axiomatised by:

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

The translation from $\mathrm{SL}^1_{\mathrm{R}}$ to Σ^1_1 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 axiomatised the following way :

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

Now we can translate $\mathrm{SL}^1_{\mathrm{R}}$ into Σ^1_1 under *at-most-binary* trees:

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

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 $[\varphi](s)$.

The preceding translation ensures that each θ_i is equivalent, under *at-most-binary* trees, 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 $i \in I, \theta_i$ is true in any member of $T_{max(I)+1}$ and θ_∞ is false in some of these models. 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. If Θ' is in $\mathrm{SL}^1_{\mathrm{R}}$ then, by the compactness theorem, $\{\mathrm{BT}(R, R^*, r)\} \cup \{\theta_i\}_{i\in\mathbb{N}} \cup \{\neg\Theta'\}$ is satisfiable, which is a contradiction. So Θ' is not SAT-equivalent to a first-order formula under *at-most-binary* trees, 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 Prop. 4.

Now we can generalise this result to SL:

Lemma 5. *If sustainable is expressible in SL over NATSs, then it is expressible in SL^1 over NATSs.*

Proof. Let Ag be a set of agent and p a proposition, and let $[\mathcal{M}_{Ag}]$ be the class of NATSs with set of agents Ag. We also call SL^{Ag} the SL language over Ag. Let a be an agent in Ag and suppose there is a formula $\varphi_{Ag}[a] \in \mathrm{SL}^{Ag}$ expressing the sustainable control of a over p. So $\varphi_{Ag}[a]$ is such that for all $\mathcal{M} \in [\mathcal{M}_{Ag}]$, for all state s in its domain, $\mathcal{M}, s \models_{\mathrm{NATS}} \varphi_{Ag}[a]$ iff a sustainably controls p from state s. We call $\mathrm{SC}_{Ag}(a,p)$ the relative class of structures. Now, let $[\mathcal{M}_{Ag\mapsto a}]$ be the class of models with set of agents Ag and such that the transitions only depend on choices made by a (for all $a' \in Ag\backslash\{a\}$, for all $s \in M$, $Ch(a', s) = M$). It is easy to see that $[\mathcal{M}_{Ag\mapsto a}] \cap \mathrm{SC}_{Ag}(a,p)$ is not empty and not equal to $[\mathcal{M}_{Ag\mapsto a}]$ (models $\mathcal{M}_1$ and $\mathcal{M}_2$ from Fig. 2 induce respective examples for proposition *access* and agent *server*).

Now, let $\mathcal{M}_{Ag} = \langle Ag, M, At, v, Ch_{Ag}\rangle \in [\mathcal{M}_{Ag\mapsto a}]$. We define $\mathcal{M}_a \in [\mathcal{M}_{\{a\}}]$ so that the choices of for a are the same in $\mathcal{M}_{Ag}$ as in $\mathcal{M}_a$: $\mathcal{M}_a = \langle Ag, M, At, v, Ch_a\rangle$ where, for all $s \in M$, $Ch_a(a, s) = Ch_{Ag}(a, s)$.

One checks that, the other way, for each $\mathcal{M}_a = \langle Ag, M, At, v, Ch_a\rangle \in [\mathcal{M}_{\{a\}}]$ there is $\mathcal{M}_{Ag} = \langle Ag, M, At, v, Ch_{Ag}\rangle \in [\mathcal{M}_{Ag\mapsto a}]\rangle \in [\mathcal{M}_{Ag\mapsto a}]$ such that for all $s \in M$, $Ch_a(a, s) = Ch_{Ag}(a, s)$ and for all $a' \in Ag\backslash\{a\}$, for all $s \in M$, $Ch(a', s) = M$.

Now, let $\varphi[a]$ be the formula obtained from $\varphi_{Ag}[a]$ by deleting every binder (a', x) such that $a' \neq a$.

Again we note let $\theta_\infty := \langle\langle x\rangle\rangle(a \rhd x)\Box(\langle\langle y\rangle\rangle(a \rhd y)\mathbf{X}p \wedge \langle\langle y\rangle\rangle(a \rhd y)\mathbf{X}\neg p)$ again. For all $\mathcal{M}_{Ag} \in [\mathcal{M}_{Ag}], \mathcal{M}_{Ag} \models_{\text{NATS}} \varphi_{Ag}[a]$ iff $\mathcal{M}_{Ag} \models \theta_{infty}$. Furthermore $\mathcal{M}_{Ag} \models_{\text{NATS}} \varphi_{Ag}[a]$ iff $\mathcal{M}_a \models_{\text{NATS}} \varphi[a]$ and $\mathcal{M}_{Ag} \models \theta_\infty$ iff $\mathcal{M}_a \models \theta_\infty$.

Eventually, for all $\mathcal{M}_a \in [\mathcal{M}_{\{a\}}], \mathcal{M}_a \models \theta_\infty$ iff $\mathcal{M}_a \models_{\text{NATS}} \varphi[a]$. Then, $\varphi[a] \in \text{SL}^1$ expresses sustainable control of a over p in SL^1, in contradiction with Prop. 4. □

Thus, sustainable control is not expressible in SL, which achieves the proof of Prop. 3.

4. Model-Checking

In this section we discuss the model-checking of USL and USL^0. 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^0 is much more tractable, since we get Pspace completeness.

4.1. USL.

As for SL, 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 the model-checking.

Theorem 1. *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] formula φ at state s of a CGS $\mathcal{M}$. It is k-ExpSpace-hard. By Prop. 1 it reduces to the problem MC-USL$(\mathcal{G}_\mathcal{A}, s, \overline{\varphi})$. The transformation preserves the number of quantifier alternations. It also yields a model with domain of size $|\mathcal{M}| \times X^\Sigma$. And it yields a formula $\overline{\varphi}$ with size bounded by $3 \times |\varphi|$. So it is in linear space w.r.t. 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 yields the following theorem:

Theorem 2. *The model-checking problem for USL is NonElementary.*

Here we prove the effective existence of a non-elementary algorithm for the model-checking of USL. To do so we build, for every formula $\varphi \in$ USL, NATS $\mathcal{M}$ and state s of $\mathcal{M}$, a nondeterministic parity automaton $\mathcal{N}^s$. If φ is a sentence, it is s.t. the language of $\mathcal{N}^s$, $\mathcal{L}(\mathcal{N}^s)$, is empty if and only if $\mathcal{M}, s \models \varphi$. This proof is adapted from [4, 9, 11]. In the following paragraphs we give the necessary preliminary definitions:

4.1.1. *Trees.*

Definition 18. *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 $path_\mathcal{T} = \{s_0.s_1 \cdots \in S^\omega \mid \forall i \in \mathbb{N}, s_0.s_1 \ldots s_i \in T\}$. 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 $\mathcal{T}_i = \langle T, l_i\rangle$.

4.1.2. *Alternating-Tree Automata.* Alternating-tree automata are a generalisation 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 19. *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 20. *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,*
- *$q_0 \in Q$ is the initial state,*
- *$\tau : (Q \times \Delta) \to$PBF$(S \times Q)$ is the transition function,*
- *$Acc : Q^\omega \to \{\top, \bot\}$ is an* acceptation function.

Under these notations we call S the set of directions *of automaton $\mathcal{A}$. Nondeterministic S-automata upon Δ ($\langle S, \Delta\rangle - NTAs$ for short) can be defined as particular cases of $\langle S, \Delta\rangle - ATAs$: an 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.

4.1.3. *Runs and Parity Acceptance Conditions.*

Definition 21. *Let $\mathcal{A} = \langle Q, q_0, \tau, Acc\rangle$ 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. $u = (t, q) = (t_0 t_1 \dots t_n, q_0 q_1 \dots q_n)$ we have that:*

- *$t \in T$*
- *The set $dir_u(U) = \{(s_0, q'_0), (s_1, q'_1), \dots, (s_n, q'_n)\} \subseteq S \times Q$ 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.

4.1.4. *Preliminary Lemma.* 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 contexts χ s.t. $\mathcal{M}, \chi, s \models \varphi$. The used encoding is formally defined in Def.22. $\mathcal{A}^{\mathcal{M},\varphi}$, and $\mathcal{A}^{\mathcal{M},\varphi}$ 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 $APTs$. They are described by lemma 6. The case for temporal operators $\mathbf{X}$ and $\mathbf{U}$ are treated the usual way.

In this construction, the inductive case for existential operators uses the operation of existential projection for $NTAs$. Then, passing an existential step requires the nondeterminisation of the APT. Lemma 7 enables this required operation. 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.

Lemma 6 (Intersection and complementation)**.** *[13, 14] 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 $\langle S, \Delta\rangle - APT$ $\mathcal{C}$ accepting language $A \cap B$. The size $|\mathcal{C}|$ of $\mathcal{C}$ 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 $\mathcal{A} = \langle Q, q_0, \tau, Acc\rangle$, then $\mathcal{D} = \langle Q, q_0, \overline{\tau}, \overline{Acc}\rangle$ 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)**.** *[14] 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 $id_{\mathcal{N}} = \bigcirc(|\mathcal{A}|.id_{\mathcal{A}}.log(|\mathcal{A}|))$.*

4.1.5. *Inputs of the Automata.* The inputs for the automata checking the satisfaction of formulas by a context are composed by an encoding of a context together with a state. From Def. 18, the input must be from a finite set. However, USL semantics defines choices by help of a context $\chi = (\chi, \kappa)$, where commitment κ stands for any word upon $(Ag \times X)$. 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 $\theta := (A \rhd x)\theta'$ s.t. ψ is a subformula of θ. Now, let K^k be the set of finite words over $(Ag \times X)^*$ 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 Ag to $\mathcal{P}(M)$ s.t. for every $a \in Ag$, $Dc^s(a) \in Ch(a, s)$.

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 i s.t. $d_s \leq i \leq d_{\max}$, $e_s(i) = e_s(d_s)$. We designate by D the set $[0, \ldots d_{\max}]$. Now, let us write Ac and call *actions* the set of partial functions from X to D.

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

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

Definition 22. *Let $\mathcal{M}$ be a NATS, s a state in M, χ a context 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 $T \subseteq Track$, is the* state-context Encoding *for $\chi = (\alpha, \kappa)$ iff it holds that for every*

$t \in T, l(t) = (last(t), ac, \kappa)$, where ac is s.t. $out(l(t)) = \chi(last(t))$. If T is the context-state encoding for χ, we call $Proj_{Ac \times K^j}(\mathcal{T})$ the context encoding for χ.

4.1.6. *Checking a State-Context 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-context encodings of contexts 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 $\langle M, M \times Ac \times K^0 \rangle - APT$ $\mathcal{A}^{\mathcal{M}}_{\varphi}$ s.t. for all states s of $\mathcal{M}$ and contexts χ with commitment $\in K^0$ for $\mathcal{M}$, it holds that $\mathcal{M}, \chi, s \models \varphi$ iff $\mathcal{T} \in \mathcal{L}(\mathcal{A}^{\mathcal{M}}_{\varphi})$, where $\mathcal{T}$ is the state-context encoding for χ.*

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 context $\chi \in K^{\mathrm{bd}(\psi;\varphi)}$, it holds that $\mathcal{M}, \chi, s \models \psi$ iff $\mathcal{T} \in \mathcal{L}(\mathcal{A}^{\mathcal{M}}_{\psi})$, where $\mathcal{T}$ is the state-context encoding for χ.

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 $\psi \in v(s)$ and $\tau_{\psi}(\overline{\psi}, (P, s)) = \bot$ otherwise.
- Case $\psi = \neg\psi_1$. If $\mathcal{A}^{\mathcal{M}}_{\psi_1} = \langle Q_{\psi_1}, q_{0_{\psi_1}}, \tau_{\psi_1}, Acc_{\psi_1} \rangle$, then by lemma 6, $\mathcal{A}^{\mathcal{M}}_{\psi} = \langle Q_{\psi_1}, q_{0_{\psi_1}}, \overline{\tau_{\psi_1}}, \overline{Acc_{\psi_1}} \rangle$.
- Case $\psi = \psi_1 \wedge \psi_2$. If $\mathcal{A}^{\mathcal{M}}_{\psi_1} = \langle Q_{\psi_1}, q_{0_{\psi_1}}, \tau_{\psi_1}, Acc_{\psi_1} \rangle$ and $\mathcal{A}^{\mathcal{M}}_{\psi_2} = \langle Q_{\psi_2}, q_{0_{\psi_2}}, \tau_{\psi_2}, Acc_{\psi_2} \rangle$ then $\mathcal{A}^{\mathcal{M}}_{\psi} = \langle Q_{\psi}, q_0, \tau_{\psi}, Acc_{\psi} \rangle$ 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 $ent \in \Delta$, $\tau_{\psi}(q_{0_{\psi}}, ent) = \tau_{\psi_1}(q_{0_{\psi_1}}, ent) \wedge \tau_{\psi_2}(q_{0_{\psi_2}}, ent)$
  - 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 $Acc_{\psi_1} = (F^{\psi_1}_1, \ldots F^{\psi_1}_{k_1})$ and $Acc_{\psi_2} = (F^{\psi_2}_1, \ldots F^{\psi_2}_{k_2})$, $\max(\{k_1, k_2\}) = k_i$ and $\min(\{k_1, k_2\}) = k_j$ then $Acc_{\psi} = (F^{\psi_1}_1 \cup F^{\psi_2}_1, \ldots, F^{\psi_1}_{k_j} \cup F^{\psi_2}_{k_j}, F^{\psi_i}_{k_j+1}, \ldots F^{\psi_i}_{k_i-1}, Q_{\psi})$.
- 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 $\mathcal{A}^{\mathcal{M}}_{\psi_1} = \langle Q_{\psi_1}, q_{0_{\psi_1}}, \tau_{\psi_1}, Acc_{\psi_1} \rangle$ then $\mathcal{A}^{\mathcal{M}}_{\psi} = \langle Q_{\psi}, q_{0_{\psi}}, \tau_{\psi}, Acc_{\psi} \rangle$ where:
  - $Q_{\psi} = \{q_{o_{\psi}}\} \cup Q_{\psi_1}$ and $q_o \notin Q_{\psi_1}$
  - for every $ent \in \Delta, \tau_{\psi}(q_{0_{\psi}}, ent) = \bigwedge_{s \in out(ent)}(s, q_{0_{\psi}})$
  - for every $q \in Q_{\psi} \neq q_{0_{\psi}}$ and for every $ent \in \Delta, \tau_{\psi}(q, ent) = \tau_{\psi_1}(q, ent)$
  - If $Acc_{\psi_1} = (F^{\psi_1}_1, \ldots F^{\psi_1}_{k_1})$ then $Acc_{\psi} = (F^{\psi_1}_1, \ldots F^{\psi_1}_{k_1} \cup \{q_{0_{\psi}}\})$
- 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 $q_{0_{\psi}}$, $q_{0_{\psi}}$ is included in the first set of the parity acceptance condition. Precisely, if $\mathcal{A}^{\mathcal{M}}_{\psi_1} = \langle Q_{\psi_1}, q_{0_{\psi_1}}, \tau_{\psi_1}, Acc_{\psi_1} \rangle$ and $\mathcal{A}^{\mathcal{M}}_{\psi_2} = \langle Q_{\psi_2}, q_{0_{\psi_2}}, \tau_{\psi_2}, Acc_{\psi_2} \rangle$ then $\mathcal{A}^{\mathcal{M}}_{\psi} = \langle Q_{\psi}, q_0, \tau_{\psi}, Acc_{\psi} \rangle$ 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 $ent \in \Delta$, $\tau_{\psi}(q_{0_{\psi}}, ent) = \tau_{\psi_2}(q_{0_{\psi_2}}, ent) \vee (\tau_{\psi_1}(q_{0_{\psi_1}}, ent \wedge \bigwedge_{s \in out(ent)}(s, q_{0_{\psi}}))$
  - 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 $Acc_\psi = (\{q_{0_\psi}\} \cup F_1^{\psi_1} \cup F_1^{\psi_2}, \dots, \{q_{0_\psi}\} \cup F_{k_j}^{\psi_1} \cup F_{k_j}^{\psi_2}, \{q_{0_\psi}\} \cup F_{k_j+1}^{\psi_i}, \dots \{q_{0_\psi}\} \cup F_{k_i-1}^{\psi_i}, Q_\psi)$.

- The case for $\psi = (A \rhd x)\psi_1$ 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 $\mathcal{A}_{\psi_1}^{\mathcal{M}} = \langle Q_{\psi_1}, q_{0_{\psi_1}}, \tau_{\psi_1}, Acc_{\psi_1}\rangle$, then $\mathcal{A}_{\psi}^{\mathcal{M}} = \langle Q_{\psi_1}, q_{0_{\psi_1}}, \tau_{\psi}, Acc_{\psi_1}\rangle$ where for every $q \in Q_{\psi_1}$ and for every $(s, ac, \kappa) \in \Delta, \tau_{\psi_1}(q, (s, ac, \kappa)) = \tau_{\psi}(q, (s, ac, \kappa[A \to x])))$.
- The case for $\psi = (A \not\rhd x)\psi_1$ 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 $\mathcal{A}_{\psi_1}^{\mathcal{M}} = \langle Q_{\psi_1}, q_{0_{\psi_1}}, \tau_{\psi_1}, Acc_{\psi_1}\rangle$, then $\mathcal{A}_{\psi}^{\mathcal{M}} = \langle Q_{\psi_1}, q_{0_{\psi_1}}, \tau_{\psi}, Acc_{\psi_1}\rangle$ where, for every $q \in Q_{\psi_1}$ and for every $(s, ac, \kappa) \in \Delta, \tau_{\psi_1}(q, (s, ac, \kappa)) = \tau_{\psi}(q, (s, ac, \kappa[A \nrightarrow x]))$.
- For the case $\psi = \langle\!\langle x \rangle\!\rangle \psi$, the transition function of the automaton for ψ 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 nondeterminise the automaton for ψ_1. If $\mathcal{A}_{\psi_1}^{\mathcal{M}} = \langle Q_{\psi_1}, q_{0_{\psi_1}}, \tau_{\psi_1}, Acc_{\psi_1}\rangle$ is an $\langle M, \Delta\rangle - APT$, then by lemma 7, there is an $\langle M, \Delta\rangle - NPT$ $\mathcal{A'}_{\psi_1}^{\mathcal{M}} = \langle Q'_{\psi_1}, q'_{0_{\psi_1}}, \tau'_{\psi_1}, Acc'_{\psi_1}\rangle$ accepting the same language, and s.t. $|\mathcal{A'}_{\psi_1}^{\mathcal{M}}| = 2^{\bigcirc(|\mathcal{A}_{\psi_1}^{\mathcal{M}}|.id_{\mathcal{A}_{\psi_1}^{\mathcal{M}}}.log(|\mathcal{A}_{\psi_1}^{\mathcal{M}}|))}$ and $id_{\mathcal{A'}_{\psi_1}^{\mathcal{M}}} = \bigcirc(|\mathcal{A}_{\psi_1}^{\mathcal{M}}|.id_{\mathcal{A}_{\psi_1}^{\mathcal{M}}}.log(|\mathcal{A}_{\psi_1}^{\mathcal{M}}|))$. For the projection we define the $\langle M, \Delta\rangle - NPT \mathcal{A}_{\psi}^{\mathcal{M}} = \langle Q'_{\psi_1}, q'_{0_{\psi_1}}, \tau_{\psi}, Acc'_{\psi_1}\rangle$ where for every $(s, ac, \kappa) \in \Delta, \tau_{\psi}(q, (s, ac, \kappa)) = \bigvee_{Dc^s \in DC^s} \tau'_{\psi_1}(q, (s, ac_{dcs}, \kappa)$, where $ac_{dcs} = ac[x \to Dc^s]$.

□

4.1.7. *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-context encoding is coherent with the node itself:

Lemma 9 (direction projection [11])**.** *Let $\mathcal{N}$ be an $\langle M, M \times \Delta\rangle$-$NPT$ and $s_0 \in M$. Then there is an $\langle M, \Delta\rangle$-NPT $\mathcal{N}^{s_0}$ s.t. for every Δ-labelled M-tree $\mathcal{T} = \langle T, v\rangle$, $\mathcal{T} \in \mathcal{L}(\mathcal{N}^{s_0})$ 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. $v'(t) = (last(s_0.t), v(t))$, for every $t \in T$. Moreover, $|\mathcal{N}^{s_0}| = |\Delta|.|\mathcal{N}|$ and $id_{\mathcal{N}^{s_0}} = id_{\mathcal{N}}$.*

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 context χ for $\mathcal{M}$, the problem MC-USL $(\mathcal{M}, s, \chi, \varphi)$ of model-checking formula φ in state s and context χ for $\mathcal{M}$ reduces to the question whether $\mathcal{T} \in \mathcal{L}(\mathcal{A}_{\varphi}^{\mathcal{M}})$, where $\mathcal{T}$ is the state-context encoding for χ. Furthermore, $\mathcal{A}_{\varphi}^{\mathcal{M}}$ is a $\langle M, M \times Ac \times K^0\rangle - APT$. By lemma 7, there is a $\langle M, Ac \times K^0\rangle - NPT$ $\mathcal{N}_{\varphi}^{\mathcal{M}}$ accepting the same language as $\mathcal{A}_{\varphi}^{\mathcal{M}}$, and s.t. $|\mathcal{N}_{\varphi}^{\mathcal{M}}| = 2^{\bigcirc(|\mathcal{A}_{\varphi}^{\mathcal{M}}|.id_{\mathcal{A}_{\varphi}^{\mathcal{M}}}.log(|\mathcal{A}_{\varphi}^{\mathcal{M}}|))}$ and $id_{\mathcal{N}_{\varphi}^{\mathcal{M}}} = \bigcirc(|\mathcal{A}_{\varphi}^{\mathcal{M}}|.id_{\mathcal{A}_{\varphi}^{\mathcal{M}}}.log(|\mathcal{A}_{\varphi}^{\mathcal{M}}|))$.

By lemma 9 applied on $\mathcal{N}_{\varphi}^{\mathcal{M}}$, we have an $\langle M, Ac \times \kappa^0\rangle$-$NPT$ $\mathcal{N}_{\varphi}^{\mathcal{M},s}$ accepting exactly encodings of contexts χ s.t. $\mathcal{M}, \chi, s \models \varphi$.

One easily checks that, φ being a sentence, for every context χ with empty context, $\mathcal{M}, \chi, s \models \varphi$ iff and only $\mathcal{M}, \chi_\emptyset, \kappa_\emptyset, s \models \varphi$. Then $\mathcal{M}, s \models \varphi$ iff $\mathcal{L}(\mathcal{N}_{\varphi}^{\mathcal{M},s})$ is not empty. The emptiness problem for

nondeterministic automata with n states and index k is solvable in $\bigcirc(n^h)$ [10]. Let us note $|\varphi|$ the length of formula φ. Each step of building for $\mathcal{N}_\varphi^{\mathcal{M}}$ 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 $\mathcal{N}_\varphi^{\mathcal{M}}$ to $\mathcal{N}_\varphi^{\mathcal{M},s}$ increases its size by factor $|\mathcal{M}|$. Then $\mathcal{N}_\varphi^{\mathcal{M},s}$ 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:

- $et(0,n_2,n_3) = n_2$
- 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 2, the model-checking for USL is decidable in time $\bigcirc(|\mathcal{M}|.et(m,2,m))$. □

4.2. **USL^0.** Here we present results of PSPACE-completeness for the model-checking of USL^0:

4.2.1. *MC-USL ^{0} is PSPACE-hard.* This result is obtained by reduction of QBFSAT$_k$ to MC-USL ^{0}, for every $k \in \mathbb{N}$. The proof is inspired by [3, 4].

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

Proof. For $k \in \mathbb{N}$, we consider the Σ_k-hard problem 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 $\mathcal{G}_\mathcal{I} = \langle Ag_I, M_I, At_I v_I, Ch_I\rangle$ 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}:

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

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:

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

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 $\langle\langle x_l\rangle\rangle(a_l \rhd x)\Diamond\psi$ (resp. $\neg\langle\langle x_l\rangle\rangle(a_l \rhd x)\Diamond\psi$) 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:

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

where for any integer l such that $1 \leq l \leq k$, $\neg^l$ 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 QBFSAT$_k$. Then for every $k \in \mathbb{N}$, QBFSAT$_k$ reduces to the model-checking of the fragment of USL^0 with k

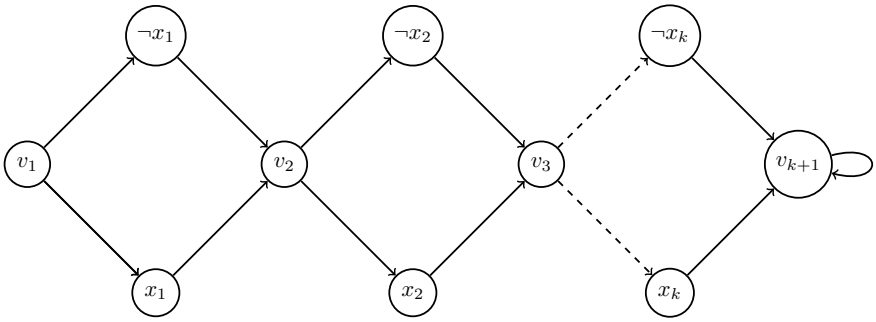

FIGURE 3. $\mathcal{G}_{\mathcal{I}}$: an illustration of QBFSAT$_k$ as a problem of model- checking for memory-less USL

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

□

4.2.2. *MC-USL ^{0} is PSPACE-Complete.* The algorithm for MC-USL ^{0} 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.

Definition 23 ($\mathcal{M}_\chi$). *Let $\mathcal{M} = \langle Ag, M, At, v, Ch\rangle$ be a NATS and let $\chi = (\alpha, \kappa)$ be a context over memory-less strategies in $\mathcal{M}$. Then $\mathcal{M}_\chi = \langle Ag, M, At, v, Ch_\chi\rangle$ is the NATS defined by: for every $(a, s) \in Ag \times M$, $Ch_\chi(a, s) = \{\chi(s) \cap c$ iff it is not empty, else $\chi(a, s)\}_{c\in Ch(a,s)}$.*

Now, we introduce the notations enabling to consider a path formula as an LTL formula over its *state subformulas* (that is its subformulas of type $\langle\langle x\rangle\rangle\psi$, $(a \rhd x)\psi$ or $(a \not\rhd x)\psi$) . For a formula φ, let us write $\mathcal{Q}(\varphi)$ the set of its outermost state subformulas , and LTLT$_\varphi$ the formula obtained from φ by replacing in it every subformula ψ in $\mathcal{Q}(\varphi)$ 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.

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.

5. RELATED WORK

Several directions have already been explored for extensions of ATL-ATL* considering the strategies played by different coalitions of agents. Table 1 sums up the main mechanisms at stake in this article and their occurrences in related works.

In the logic ATLES [16], 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. These predefined strategy terms are interpreted as semantic strategies, and the fact to refer to strategies in the syntax is not convenient in practice. This logic has a PTIME model-checking. However, its expressiveness is strictly higher that ATL but incomparable with ATL*.

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

Require: A NATS $\mathcal{M}$, a context $\chi = (\kappa, \chi)$, $s_0 \in M$ and an USL path formula φ
Ensure: YES iff $\mathcal{M}, \chi, s \models \varphi$ $\mathcal{M}' = \mathcal{M}_\chi$

```
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, χ[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], χ), s, ψ) then
        label s' with ψ̄
      end if
    else if ψ = (A ▷̸ x)ψ' then
      if MC-USL (M', (κ[A ↛ x], χ), s, ψ) then
        label s' with ψ̄
      end if
    end if
  end for
end for
return MC-LTL(M', s, LTLT_φ)
```

In BSIL [17], an operator explicitly mentions that a strategy bound to a coalition can be composed with the context. In terms of expressiveness, it extends ATL but is incomparable with ATL*. It is subsumed by SL and ATL_{sc} (and therefore by USL) but has "only" a PSPACE-complete model-checking problem.

The overwriting of strategies is also questioned in IATL [1]. In this proposition, the authors make a distinction between ATL- and SL-style revocable strategies and irrevocable strategies. They propose a formalism for the latter. We believe that USL strategies offer an adequate synthesis between both views because they can be modified later and hold, at the same time, some definitive commitments from agents.

The work presented here deeply refers to SL [8, 11, 12], which fully enables to compose the different strategies followed by agents in a context. Nevertheless, the composition of several strategies for one agent is not possible in that formalism, since an agent overwrites her previous strategy when she is bound to a new one.

Incidentally, SL also bears a syntactic constraint that is not present in USL: all agents must be explicitly bound to a strategy before evaluating a temporal formula. In addition to the unnecessary important size of formulas, this raises the more fundamental problem of (lack of) *modularity*: a single informal specification may give rise to *different* SL formulas depending on the number of agents of the system not mentioned in the specification.

The idea of agents explicitly unbound from their current strategies is also present in ATL_{sc} [4, 9] 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 her strategy.

All these formalisms enrich the composition of strategies defined in ATL with a notion of strategy context. But they only compose strategies if they concern distinct agents. We figure USL as a further step

TABLE 1. Strategy contexts, revocation and refinement in multi-agent temporal logics (p stands for "partial support").

	Strategy contexts	Explicit revocation	No systematic revocation	Revocable strategies	Refinable strategies	Sustainable capabilities
ATL				×		
ATLES	p	p		×		
BSIL	p	p		×		
IATL	p		×			
SL	×			×		
ATL_{sc}	×	×		×		
USL	×	×	×	×	×	×

in this contextualisation. In USL, binding an agent to a strategy commits her to play this strategy. This strategy can then be refined, and its possible revocation must be explicit.

6. 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. The syntax of USL is also more flexible than that of SL, and is better adapted to modular specifications. Its model-checking problem is NONELEMENTARY but it is PSPACE-complete for its memory-less restriction.

As future work, we plan to study applications of USL. In particular, as we already noticed in [5], temporal multi-agents logics can be useful to investigate goal- and agent-oriented requirements engineering problems. We will investigate on the possibilities offered by USL for the verification of requirements engineering models.

We will also 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 analysed.

Finally, we want to compare USL with the logic QD_μ [15]. The latter formalism enables to express fixed-point properties about strategies and subsumes SL. Since the concept of sustainable capabilities, which seems to characterise well USL expressiveness, is close to a fixed point, we think that this comparison may yield interesting results.

REFERENCES

[1] Ågotnes, T., Goranko, V., Jamroga, W.: Alternating-time temporal logics with irrevocable strategies. In: Theoretical aspects of rationality and knowledge. pp. 15–24 (2007)
[2] Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM 49(5), 672–713 (2002)
[3] Baier, C., Brazdil, T., Grosser, M., Kucera, A.: Stochastic game logic. In: Quantitative Evaluation of Systems. pp. 227 –236 (2007)
[4] Brihaye, T., Da Costa Lopes, A., Laroussinie, F., Markey, N.: ATL with strategy contexts and bounded memory. Logical Foundations of Computer Science pp. 92–106 (2009)
[5] Chareton, C., Brunel, J., Chemouil, D.: A formal treatment of agents, goals and operations using alternating-time temporal logic. In: Brazilian Symposium on Formal Methods (SBMF). pp. 188–203 (2011)

[6] Chareton, C., Brunel, J., Chemouil, D.: Vers une sémantique des jeux pour un langage d'ingénierie des exigences par buts et agents. In: Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL) (2012), http://hal.archives-ouvertes.fr/hal-00782773
[7] Chareton, C., Brunel, J., Chemouil, D.: Towards an Updatable Strategy Logic. In: Proc. 1st International Workshop on Strategic Reasoning SR (2013), http://arxiv.org/abs/1303.0795
[8] Chatterjee, K., Henzinger, T.A., Piterman, N.: Strategy logic. Inf. & Comp. 208(6), 677–693 (2010)
[9] Da Costa Lopes, A.: Propriétés de jeux multi-agents. Phd thesis, École normale supérieure de Cachan (Sep 2011)
[10] Kupferman, O., Vardi, M.Y.: Weak alternating automata and tree automata emptiness. In: ACM Symposium on Theory of computing. pp. 224–233 (1998)
[11] Mogavero, F., Murano, A., Perelli, G., Vardi, M.Y.: Reasoning about strategies: On the model-checking problem. CoRR abs/1112.6275 (2011)
[12] Mogavero, F., Murano, A., Vardi, M.Y.: Reasoning about strategies. In: IARCS Annual Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS). vol. 8, pp. 133–144 (2010)
[13] Muller, D.E., Schupp, P.E.: Alternating automata on infinite trees. Theor. Comp. Sci. 54(2-3), 267–276 (Oct 1987)
[14] Muller, D.E., Schupp, P.E.: 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)
[15] Pinchinat, S.: Quantified mu-calculus with decision modalities for concurrent game structures. Tech. rep., Dept. of Computer Science, Faculty of Engineering and Information Technology, Australian National University (2007)
[16] Walther, D., van der Hoek, W., Wooldridge, M.: Alternating-time temporal logic with explicit strategies. In: Theoretical aspects of rationality and knowledge (TARK 07). pp. 269–278. ACM (2007)
[17] Wang, F., Huang, C.H., Yu, F.: A temporal logic for the interaction of strategies. In: Katoen, J.P., König, B. (eds.) CONCUR 2011 – Concurrency Theory. Lecture Notes in Computer Science, vol. 6901, pp. 466–481. Springer (2011)

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