Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020)
Main Track

Nondeterministic Strategies and their Refinement in Strategy Logic

Giuseppe De Giacomo[1] , **Bastien Maubert**[2] and **Aniello Murano**[2]
[1]Sapienza Università di Roma
[2]Università degli Studi di Napoli Federico II
degiacomo@diag.uniroma1.it, bastien.maubert@gmail.com, murano@na.infn.it

Abstract

Nondeterministic strategies are strategies (or protocols, or plans) that, given a history in a game, assign a set of possible actions, all of which should be winning. An important problem is that of refining such strategies. For instance, given a nondeterministic strategy that allows only safe executions, refine it to, additionally, eventually reach a desired state of affairs. We show that strategic problems involving strategy refinement can be solved elegantly in the framework of Strategy Logic (SL), a very expressive logic to reason about strategic abilities. Specifically, we introduce an extension of SL with nondeterministic strategies and an operator expressing strategy refinement. We show that model checking this logic can be done at no additional computational cost with respect to standard SL, and can be used to solve a variety of problems such as synthesis of maximally permissive strategies or maximally permissive Nash equilibria.

1 Introduction

Program synthesis is a fundamental problem in different areas of computer science such as formal methods, artificial intelligence or control theory. The objects to be synthesized are sometimes called systems, plans, strategies, protocols or controllers, but essentially they are always functions that define which transition to take, or which action to perform, after a given finite run of the system. The basic problem is, in essence, to synthesize (if possible) a finite representation of such a function such that the set of generated runs satisfies a given specification. When the system is *closed*, i.e., when the evolution of the system depends solely on its actions, a plan is essentially a sequence of actions, and synthesis is usually rather easy. For *open systems* that interact with an environment in a game-like manner, the synthesized program has to enforce the specification no matter how the environment behaves; it is thus no longer a sequence of actions but a tree-like object (that we will call strategy), and the synthesis problem is more challenging. Variants of this problem include supervisory control (Ramadge and Wonham 1987), reactive synthesis (Pnueli and Rosner 1989) and nondeterministic planning (Geffner and Bonet 2013).

One important difference between the different variants of the problem is that some consider *deterministic* strategies while others allow for *nondeterministic* ones, which prescribe for each finite run a set of actions to choose from nondeterministically. In particular this is the case in synthesis with reactive environments (Kupferman et al. 2000), where the environment can use nondeterministic strategies, and in supervisory control (Ramadge and Wonham 1987) where the synthesized controller not only is nondeterministic, but also should allow as many behaviors as possible without breaking the specification. Indeed, in the classic setting that considers only safety objectives, the existence of a maximally permissive controller is always guaranteed (Wonham 2014). Note that it is the only type of objectives for which this holds (Bernet, Janin, and Walukiewicz 2002).

The notion of maximal permissiveness is important also because a more permissive system can more easily be *refined* later on to make it satisfy additional requirements. For instance, given a nondeterministic strategy that allows only safe executions, one may want to refine it to, in addition, be sure to eventually reach a desired state of affairs. Maximally permissive strategies have thus been studied in generalizations of supervisory control with specifications that go beyond safety (Pinchinat and Riedweg 2005), but also in computational game theory in the formal methods community (Bernet, Janin, and Walukiewicz 2002; Bouyer et al. 2011) and in the community of reasoning about actions (De Giacomo, Lespérance, and Muise 2012; De Giacomo, Patrizi, and Sardiña 2013; Banihashemi, De Giacomo, and Lespérance 2018).

The literature thus contains a plethora of different synthesis problems. To provide a general framework to specify and solve such problems, Strategy Logic was introduced in (Chatterjee, Henzinger, and Piterman 2010), and extended to the multi-agent setting in (Mogavero et al. 2014). This very expressive logic treats strategies as first-order objects, and can express a variety of complex synthesis problems such as distributed synthesis, synthesis of Nash equilibria or rational synthesis (Čermák et al. 2018; Belardinelli et al. 2020; Berthon et al. 2020). However Strategy Logic considers only deterministic strategies, and thus cannot naturally capture problems such as supervisory control or synthesis with reactive environments.

In this work we introduce an extension of Strategy Logic that allows for nondeterministic strategies and contains a *refinement operator* $\preceq$. In the resulting logic, called $\mathsf{SL}^{\prec}$, formula $x \preceq y$ means that strategy x refines strategy y, or in other words, that y is more permissive than x. Because quantification on deterministic strategies and maxi-

mal permissiveness can be expressed using quantification on nondeterministic strategies and the refinement operator, $\mathsf{SL}^{\prec}$ strictly extends SL and can capture, in addition, all the synthesis problems mentioned above. It can also express module checking (Kupferman, Vardi, and Wolper 2001), and in the context of multi-agent systems, $\mathsf{SL}^{\prec}$ can be used to synthesize maximally permissive Nash equilibria.

When the specification is expressed in a branching-time logic such as CTL^*, formulas such as $\mathbf{E}\psi_1 \wedge \mathbf{E}\psi_2$ express the existence of runs satisfying ψ_1 and others satisfying ψ_2. When both the system and the environment use nondeterministic strategies, it does not say however that the system can choose to enforce ψ_1 or ψ_2 independently of what the environment does. We show that this property, which we call *unilateral forcing*, can be expressed in $\mathsf{SL}^{\prec}$.

We solve the model-checking problem for $\mathsf{SL}^{\prec}$, and establish that it is no harder than for classic SL. As it is usually the case for this kind of logics, the model-checking algorithm for $\mathsf{SL}^{\prec}$ can provide finite witness strategies when they exist, such that we obtain a unified procedure to solve all the synthesis problems from the literature mentioned above, with optimal asymptotic complexity, and solve new ones, such as nondeterministic synthesis with unilateral forcing, of synthesis of maximally permissive Nash equilibria.

Plan. We introduce nondeterministic strategies and their refinement in Section 2. In Section 3 we present the logic $\mathsf{SL}^{\prec}$, and in Section 4 we show how it captures a number of problems, some existing and others new. The model-checking procedure for $\mathsf{SL}^{\prec}$ is presented in Section 5, and we conclude in Section 6.

2 Nondeterministic Strategies

We first recall classic concurrent game structures, nondeterministic strategies, and the notion of strategy refinement.

2.1 Notations

Let Σ be an alphabet. A *finite* (resp. *infinite*) *word* over Σ is an element of Σ^* (resp. Σ^ω). The *length* of a finite nonempty word $w = w_0 w_1 \dots w_n$ is $|w| := n + 1$, and $\mathrm{last}(w) := w_n$ is its last letter; the length of the empty word is 0. Given a finite (resp. infinite) word w and $0 \leq i < |w|$ (resp. $i \in \mathbb{N}$), we let w_i be the letter at position i in w, $w_{\leq i}$ is the prefix of w that ends at position i and $w_{\geq i}$ is the suffix that starts at position i. We write $w \preccurlyeq w'$ if w is a prefix of w'.The domain of a mapping f is written $\mathit{dom}(f)$.

2.2 Concurrent Game Structures

Let us fix AP, a finite non-empty set of *atomic propositions*, and Ag, a finite non-empty set of *agents* or *players*.

Definition 1. A *concurrent game structure* (or CGS) is a tuple $\mathcal{G} = (\mathrm{Ac}, V, E, \ell, v_\iota)$ where

- Ac is a finite non-empty set of *actions*,
- V is a finite non-empty set of *positions*,
- $E : V \times \mathrm{Ac}^{\mathrm{Ag}} \to V$ is a *transition function*,
- $\ell : V \to 2^{\mathrm{AP}}$ is a *labeling function*, and
- $v_\iota \in V$ is an *initial position*.

In a position $v \in V$, where atomic propositions $\ell(v)$ hold, each player a chooses an action $c_a \in \mathrm{Ac}$, and the game proceeds to position $E(v, \boldsymbol{c})$, where $\boldsymbol{c} \in \mathrm{Ac}^{\mathrm{Ag}}$ stands for the *joint action* $(c_a)_{a \in \mathrm{Ag}}$. Given a joint action $\boldsymbol{c} = (c_a)_{a \in \mathrm{Ag}}$ and $a \in \mathrm{Ag}$, we let $\boldsymbol{c}_a$ denote c_a. A *finite* (resp. *infinite*) *play* is a finite (resp. infinite) word $\rho = v_0 \dots v_n$ (resp. $\pi = v_0 v_1 \dots$) such that $v_0 = v_\iota$ and for every i such that $0 \leq i < |\rho| - 1$ (resp. $i \geq 0$), there exists a joint action $\boldsymbol{c}$ such that $E(v_i, \boldsymbol{c}) = v_{i+1}$. Given two finite plays ρ and ρ', we say that ρ' is a *continuation* of ρ if $\rho' \in \rho \cdot V^*$, and we write $\mathrm{Cont}(\rho)$ for the set of continuations of ρ. The size $|\mathcal{G}|$ of a CGS $\mathcal{G}$ is its number of positions.

Remark 1. Recall that turn-based game structures can be seen as a special case of concurrent game structures in which, in each position, only one player's actions are relevant (Alur, Henzinger, and Kupferman 2002).

2.3 Strategy Refinement

Given a CGS $\mathcal{G}$, a *nondeterministic strategy*, or strategy for short, for a player is a function $\sigma : \mathrm{Cont}(v_\iota) \to 2^{\mathrm{Ac}} \setminus \emptyset$ that maps each finite play in $\mathcal{G}$ to a nonempty finite set of actions that the player may choose from after this finite play. A strategy σ is *deterministic* if for every finite play ρ, $\sigma(\rho)$ is a singleton. We let *Str* denote the set of all (nondeterministic) strategies, and $\mathit{Str}^{\mathrm{d}} \subset \mathit{Str}$ the set of deterministic ones (note that these sets depend on the CGS under consideration).

Formulas of our logic $\mathsf{SL}^{\prec}$ will be evaluated at the end of a finite play ρ (which can be simply the initial position of the game), and since $\mathsf{SL}^{\prec}$ contains only *future-time* temporal operators, the only relevant part of a strategy σ when evaluating a formula after finite play ρ is its definition on continuations of ρ. We thus define the *restriction* of σ to ρ as the restriction of σ to $\rho \cdot V^*$, that we write $\sigma_{|\rho} : \mathrm{Cont}(\rho) \to 2^{\mathrm{Ac}} \setminus \emptyset$. We will then say that a strategy σ refines another strategy σ' after a finite play ρ if the first one is more restrictive than the second one on continuations of ρ. More formally:

Definition 2. Strategy σ *refines* strategy σ' after ρ, written $\sigma \preceq_\rho \sigma'$, if for every $\rho' \in \mathrm{Cont}(\rho)$, $\sigma_{|\rho}(\rho') \subseteq \sigma'_{|\rho}(\rho')$. We simply say that σ refines σ' if it refines it after the initial position v_ι, and in that case we write $\sigma \preceq \sigma'$.

3 Strategy Logic with Refinement

We introduce $\mathsf{SL}^{\prec}$, which extends SL with nondeterministic strategies, an *outcome quantifier* that quantifies over possible outcomes of a strategy profile (both are already considered in (Berthon et al. 2020)), and more importantly, a refining operator that expresses that a strategy refines another.

3.1 Syntax

In addition to the sets of propositions AP and agents Ag, we now fix Var, a finite non-empty set of *variables*.

Definition 3. The syntax of $\mathsf{SL}^{\prec}$ is defined by the following grammar:

$$\varphi := p \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x\, \varphi \mid (a, x)\varphi \mid x \preceq y \mid \mathbf{E}\psi$$
$$\psi := \varphi \mid \neg\psi \mid \psi \vee \psi \mid \mathbf{X}\psi \mid \psi \mathbf{U} \psi$$

where $p \in \mathrm{AP}$, $x, y \in \mathrm{Var}$ and $a \in \mathrm{Ag}$.

Formulas of type φ are called *state formulas*, those of type ψ are *path formulas*, and $\mathsf{SL}^{\prec}$ consists of all state formulas.

Temporal operators, $\mathbf{X}$ (next) and $\mathbf{U}$ (until), have their usual meaning. The *refinement operator* expresses that a strategy is more restrictive than another or, in other words, that it allows less behaviors: $x \preceq y$ reads as "strategy x refines strategy y". The *strategy quantifier* $\exists x$ has its usual meaning, except that it now quantifies over *nondeterministic* strategies: $\exists x\varphi$ reads as "there exists a nondeterministic strategy x such that φ holds". As usual, the *binding operator* (a, x) assigns a strategy to an agent, and $(a, x)\varphi$ reads as "when agent a plays strategy x, φ holds". Finally, the *outcome quantifier* $\mathbf{E}$ quantifies on outcomes of strategies currently in use: $\mathbf{E}\psi$ reads as "ψ holds in some outcome of the strategies currently used by the players". LTL is the fragment of SL made of path formulas that use only Boolean and temporal operators.

We use the following usual abbreviations $\top := p \vee \neg p$, $\varphi \rightarrow \varphi' := \neg\varphi \vee \varphi'$, $\mathbf{A}\psi := \neg\mathbf{E}\neg\psi$, $\mathbf{F}\varphi := \top\mathbf{U}\varphi$, $\mathbf{G}\varphi := \neg\mathbf{F}\neg\varphi$ and $\forall x\varphi := \neg\exists x\neg\varphi$.

For every formula $\varphi \in \mathsf{SL}^{\prec}$, we let *free* (φ) be the set of variables that appear free in φ, i.e., that appear out of the scope of a strategy quantifier. A formula φ is a *sentence* if *free* (φ) is empty. Finally, we let the *size* $|\varphi|$ of a formula φ be the number of symbols in φ.

3.2 Semantics

$\mathsf{SL}^{\prec}$ formulas are interpreted in a CGS, and the semantics makes use of the following additional notions.

An *assignment* $\chi : \mathrm{Ag} \cup \mathrm{Var} \rightharpoonup \mathit{Str}$ is a partial function that assigns a strategy to each player and strategy variable in its domain. For an assignment χ, player a and strategy σ, $\chi[a \mapsto \sigma]$ is the assignment of domain $\mathit{dom}(\chi) \cup \{a\}$ that maps a to σ and is equal to χ on the rest of its domain, and $\chi[x \mapsto \sigma]$ is defined similarly, where x is a variable. We write $\mathrm{Ag}(\chi)$ for $\mathit{dom}(\chi) \cap \mathrm{Ag}$, and $\mathrm{Var}(\chi)$ for $\mathit{dom}(\chi) \cap \mathrm{Var}$. An assignment is *variable-complete* for a formula $\varphi \in \mathsf{SL}^{\prec}$ if *free* $(\varphi) \subseteq \mathrm{Var}(\chi)$.

For an assignment χ and a finite play ρ, we define the *outcomes of* χ *from* ρ as the set of infinite plays that start with ρ and are then extended by letting players follow the strategies assigned by χ. Formally, we define $\mathrm{Out}(\chi, \rho)$ as the set of plays of the form $\rho \cdot v_1 v_2 \dots$ such that for all $i \geq 0$, there exists $\boldsymbol{c}$ such that for all $a \in \mathrm{Ag}(\chi)$, it holds that $\boldsymbol{c}_a \in \chi(a)(\rho \cdot v_1 \dots v_i)$ and $v_{i+1} = E(v_i, \boldsymbol{c})$, with $v_0 = \mathrm{last}(\rho)$.

Definition 4. The semantics of a state (resp. path) formula is defined on a CGS $\mathcal{G}$, an assignment χ that is variable-complete for φ, and a finite play ρ (resp. an infinite play π and an index $i \in \mathbb{N}$). The definition by mutual induction is as follows (we omit Boolean cases):

$$
\begin{array}{lll}
\mathcal{G}, \chi, \rho \models p & \text{if} & p \in \ell(\mathrm{last}(\rho)) \\
\mathcal{G}, \chi, \rho \models \exists x\varphi & \text{if} & \exists\, \sigma \in \mathit{Str} \text{ s.t. } \mathcal{G}, \chi[x \mapsto \sigma], \rho \models \varphi \\
\mathcal{G}, \chi, \rho \models (a, x)\varphi & \text{if} & \mathcal{G}, \chi[a \mapsto \chi(x)], \rho \models \varphi \\
\mathcal{G}, \chi, \rho \models x \preceq y & \text{if} & \chi(x) \text{ refines } \chi(y) \text{ after } \rho \\
\mathcal{G}, \chi, \rho \models \mathbf{E}\psi & \text{if} & \exists \pi \in \mathrm{Out}(\chi, \rho) \text{ s.t.} \\
 & & \quad \mathcal{G}, \chi, \pi, |\rho| - 1 \models \psi \\
\mathcal{G}, \chi, \pi, i \models \varphi & \text{if} & \mathcal{G}, \chi, \pi_{\leq i} \models \varphi \\
\mathcal{G}, \chi, \pi, i \models \mathbf{X}\psi & \text{if} & \mathcal{G}, \chi, \pi, i+1 \models \psi \\
\mathcal{G}, \chi, \pi, i \models \psi\mathbf{U}\psi' & \text{if} & \exists\, j \geq i \text{ s.t. } \mathcal{G}, \chi, \pi, j \models \psi' \text{ and,} \\
 & & \quad \forall\, k \text{ s.t. } i \leq k < j,\ \mathcal{G}, \chi, \pi, k \models \psi
\end{array}
$$

If φ is a sentence and $\mathcal{G}$ is a CGS with initial position v_ι, then the empty assignment $\emptyset$ is variable-complete for φ and we write $\mathcal{G} \models \varphi$ for $\mathcal{G}, \emptyset, v_\iota \models \varphi$.

We give some examples of useful notions that can be expressed in this logic.

Example 1 (Strategy equality)**.** First, it is easy to see that a strategy σ is equal to another strategy σ' if $\sigma \preceq \sigma'$ and $\sigma' \preceq \sigma$. We thus define the abbreviation

$$x = y \quad := \quad x \preceq y \wedge y \preceq x$$

We have $\mathcal{G}, \chi, \rho \models x = y$ if, and only if, $\chi(x)_{|\rho} = \chi(y)_{|\rho}$. In particular, $\mathcal{G}, \chi, v_\iota \models x = y$ if, and only if, $\chi(x) = \chi(y)$. Let also $x \neq y := \neg(x = y)$ and $x \prec y := x \preceq y \wedge x \neq y$.

Example 2 (Deterministic strategies)**.** We can also express that a strategy, or its refinement to continuations of the current finite play, is deterministic, with the following formula:

$$\mathrm{det}(x) \quad := \quad \forall y\ (y \preceq x \rightarrow x \preceq y)$$

In the following we will use the following notations to express quantification on *deterministic* strategies:

$$\exists^{\mathrm{d}}\, x\, \varphi := \exists x(\mathrm{det}(x) \wedge \varphi)$$
$$\text{and} \quad \forall^{\mathrm{d}}\, x\, \varphi := \forall x(\mathrm{det}(x) \rightarrow \varphi)$$

Note that, as usual, we have $\forall^{\mathrm{d}}\, x\, \varphi \equiv \neg\, \exists^{\mathrm{d}}\, x \neg\varphi$.

3.3 Outcomes as Strategy Refinements

Before stating our main result, we point out that the outcome quantifier $\mathbf{E}$ is tightly linked to strategy refinement. More precisely, selecting an individual outcome of an assignment χ amounts to choosing a *deterministic* strategy for each player $a \in \mathrm{Ag}$, such that for all $a \in \mathrm{Ag}(\chi)$, $\sigma_a \preceq \chi(a)$. Indeed, fixing a deterministic strategy for each player fixes a unique play, and the refinement constraint ensures that this play follows the nondeterministic strategies assigned by χ.

Lemma 1. *Let ψ be an LTL formula, $\mathcal{G}$ a CGS, ρ a finite play in $\mathcal{G}$ and χ an assignment such that $\chi(a) = \chi(x_a)$ for each $a \in \mathrm{Ag}(\chi)$. It holds that*

$$\mathcal{G}, \chi, \rho \models \mathbf{E}\psi$$
$$\textit{iff}$$
$$\mathcal{G}, \chi, \rho \models \exists^{d}_{a \in \mathrm{Ag}}\, y_a \bigwedge_{a \in \mathrm{Ag}(\chi)} y_a \preceq x_a \wedge (a, y_a)_{a \in \mathrm{Ag}} \mathbf{A}\psi$$

Notice that in the last formula, $\mathbf{A}\psi$ could be replaced by $\mathbf{E}\psi$: indeed, after each agent a has been bound to deterministic strategy y_a, there exists a unique outcome. This quantifier is therefore superfluous and could be removed, writing ψ instead of $\mathbf{A}\psi$. This lemma thus shows that, from the point of view of expressivity, having the refinement operator we could get rid of the outcome quantifier. However this makes formulas bigger and more complex to model check.

3.4 Main Result

To state precisely the complexity of model checking $\mathsf{SL}^{\prec}$ we need the notion of simulation depth, introduced in (Berthon et al. 2020). The rough idea is to count how many times one has to transform an alternating automaton into a nondeterministic one of exponential size, which can then be projected to guess a strategy x (see (Berthon et al. 2020) for details). The simulation depth $\mathrm{sd}(\varphi)$ of a formula φ is a pair (k, x) where k is the number of nested simulations needed in the automata construction for φ, and $x \in \{\mathrm{nd}, \mathrm{alt}\}$ is the type of automaton obtained (nondeterministic or alternating). We write $\mathrm{sd}_k(\varphi)$ and $\mathrm{sd}_x(\varphi)$ for, respectively, the first and second component of $\mathrm{sd}(\varphi)$. It is defined inductively as follows:

$$\mathrm{sd}(p) := (0, \mathrm{nd}) \qquad \mathrm{sd}(\neg\varphi) := (\mathrm{sd}_k(\varphi), \mathrm{alt})$$

$$\mathrm{sd}(\varphi_1 \vee \varphi_2) := \left(\max_{i\in\{1,2\}} \mathrm{sd}_k(\varphi_i), x\right),$$

$$\text{where } x = \begin{cases} \mathrm{nd} & \text{if } \mathrm{sd}_x(\varphi_1) = \mathrm{sd}_x(\varphi_2) = \mathrm{nd} \\ \mathrm{alt} & \text{otherwise} \end{cases}$$

$$\mathrm{sd}(\exists x\varphi) := (k, \mathrm{nd}),$$

$$\text{where } k = \begin{cases} \mathrm{sd}_k(\varphi) & \text{if } \mathrm{sd}_x(\varphi) = \mathrm{nd} \\ \mathrm{sd}_k(\varphi) + 1 & \text{otherwise} \end{cases}$$

$$\mathrm{sd}((a, x)\varphi) := \mathrm{sd}(\varphi) \qquad \mathrm{sd}(x \preceq y) := (0, \mathrm{nd})$$

$$\mathrm{sd}(\mathbf{E}\psi) := \begin{cases} (0, \mathrm{nd}) & \text{if } \psi \in \mathsf{LTL} \\ (\max_{\varphi\in\max(\psi)} \mathrm{sd}_k(\varphi), \mathrm{alt}) & \text{otherwise} \end{cases}$$

The case for $x \preceq y$ is new, and its definition reflects the fact that this property can be checked with a simple deterministic automaton (see the translation to QCTL^* in Section 5.1). We say that a formula φ has simulation depth k if $\mathrm{sd}_k(\varphi) = k$. We now state the following result, which is proved in Section 5.

Theorem 2. *Model checking $\mathsf{SL}^{\prec}$ is $(k+1)$-EXPTIME-complete for formulas of simulation depth at most k.*

Remark 2. Defining $\exists^{\mathrm{d}} x\, \varphi$ as $\exists x\, (\det(x) \wedge \varphi)$, where $\det(x) = \forall y\, (y \preceq x \rightarrow x \preceq y)$, introduces a simulation between $\exists x$ and $\forall y$. This exponential can be avoided by considering $\exists^{\mathrm{d}} x\, \varphi$ as a basic construct in the syntax, whose translation to QCTL^* is similar to that of $\exists x\, \varphi$ (see Section 5.1), and whose simulation depth is defined as for $\exists x\, \varphi$.

In the following section we show how $\mathsf{SL}^{\prec}$ captures a number of important problems related to nondeterministic strategies and strategy refinement.

4 Applications of $\mathsf{SL}^{\prec}$

In this section we show how our framework captures generalizations of the classical LTL synthesis problem to the context of nondeterministic strategies.

4.1 Reactive Synthesis

We first recall the standard LTL synthesis problem as defined in (Pnueli and Rosner 1989): consider a set of *input variables* I controlled by the environment and a set of *output variables* O controlled by the system. In each round, first the environment chooses a valuation of the inputs $i_k \in 2^I$ (called *input*), and then the system reacts by choosing a valuation on the output variables $o_k \in 2^O$ (called *output*); an infinite word over $2^{I\cup O}$ is called an *execution*. The system has perfect recall, meaning that its choices can depend on all previous choices of the environment, and a strategy for the system is thus a function $S : (2^I)^+ \to 2^O$. Given an infinite sequence of inputs $w = i_0 i_1 i_2 \ldots \in (2^I)^\omega$, we define the execution $S(w) = i_0 \cup o_0, i_1 \cup o_1, i_2 \cup o_2 \ldots$ where, for each $k \geq 0$, $o_k = S(i_0 \ldots i_k)$.

The LTL *synthesis problem* consists in, given an LTL formula ψ over atoms $I \cup O$, synthesizing a (finite representation of a) system $S : (2^I)^+ \to 2^O$ such that for all $w = i_0 i_1 i_2 \ldots \in (2^I)^\omega$ it holds that $S(w) \models \psi$. This problem can be coded in Strategy Logic: one builds a turn-based game arena $\mathcal{G}_{I,O}$ (which can be represented as a CGS, see Remark 1) with two players, E (for Environment) and S (for System) in which first the environment chooses an input i, then the system chooses an output o, reaching a position labeled with atoms $i \cup o$ and in which it is the environment's turn to play. The LTL synthesis problem for (I, O, ψ) can then be solved by model-checking on $\mathcal{G}_{I,O}$ the $\mathsf{SL}^{\prec}$ formula

$$\varphi^{\mathrm{d}}_{\mathrm{synth}}(\psi) := \exists^{\mathrm{d}}\, x.\, \forall^{\mathrm{d}}\, y.(S, x)(E, y)\mathbf{A}\psi$$

Note that this really solves the synthesis problem as existing model-checking algorithms for Strategy Logic can synthesize witness strategies (when they exist) for strategy variables existentially quantified at the beginning of the formula. Rewriting $\varphi^{\mathrm{d}}_{\mathrm{synth}}(\psi)$ as $\exists^{\mathrm{d}}\, x.\neg\, \exists^{\mathrm{d}}\, y.(S, x)(E, y)\mathbf{E}\neg\psi$ we see that it has simulation depth 1 (see Remark 2) and thus can be solved by the model-checking algorithm for $\mathsf{SL}^{\prec}$ in doubly exponential time (Theorem 2), which is optimal since LTL synthesis is 2EXPTIME-complete (Pnueli and Rosner 1989).

Note also that in the case of deterministic strategies, fixing a strategy for each player fixes a unique outcome, and thus $\mathbf{A}\psi$ in the formula above could be replaced by $\mathbf{E}\psi$ without affecting the semantics. Also, once a deterministic strategy x is fixed for S, each deterministic strategy y for E fixes an outcome of strategy x, and each outcome of x can be obtained by fixing a deterministic strategy y for E. It then follows by the semantics of the outcome quantifier $\mathbf{A}$ that, when considering only deterministic strategies, $\varphi^{\mathrm{d}}_{\mathrm{synth}}(\psi)$ is equivalent to $\exists^{\mathrm{d}}\, x(S, x)\mathbf{A}\psi$.

4.2 Nondeterministic Synthesis

Considering nondeterministic strategies, as we do, does not change anything for classical LTL synthesis. Indeed, consider the following formula:

$$\varphi^{\mathrm{nd}}_{\mathrm{synth}}(\psi) := \exists x.\forall y.(S, x)(E, y)\mathbf{A}\psi$$

which differs from $\varphi^{\mathrm{d}}_{\mathrm{synth}}(\psi)$ only in that it now allows for nondeterministic strategies. It is easy to check that:

Proposition 3. *For every LTL formula ψ,*

$$\mathcal{G}_{I,O} \models \varphi^{d}_{synth}(\psi) \quad \textit{iff} \quad \mathcal{G}_{I,O} \models \varphi^{nd}_{synth}(\psi)$$

However it makes a difference if, instead of considering only universal satisfaction of an LTL formula on all outcomes, we consider other forms of branching-time specifications, in particular specifications that require existence of

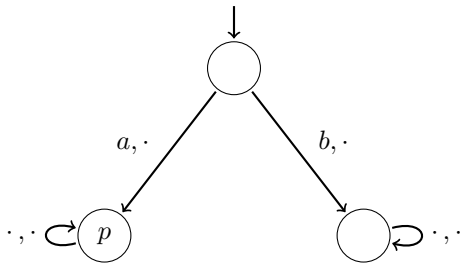

Figure 1: Model $\mathcal{G}_1$, that satisfies φ_1' and φ_2

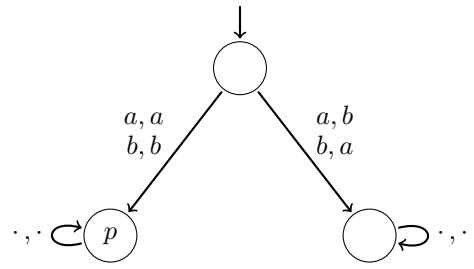

Figure 2: Model $\mathcal{G}_2$, that satisfies φ_1' but not φ_2

outcomes satisfying different properties, sometimes called *possibility requirements* (Kupferman and Vardi 1997).

For instance,

$$\varphi_1 := \exists^{\mathrm{d}}\, x(S,x)\, \forall^{\mathrm{d}}\, y(E,y)(\mathbf{EG}p \wedge \mathbf{EF}\neg p)$$

is always false, because a pair of deterministic strategies for the system and the environment determine a unique outcome that cannot satisfy both $\mathbf{G}p$ and $\mathbf{F}\neg p$. However

$$\varphi_1' := \exists x(S,x)\forall y(E,y)(\mathbf{EG}p \wedge \mathbf{EF}\neg p)$$

can be true: for instance in model $\mathcal{G}_1$ depicted in Figure 1, where the initial position is marked by an incoming arrow and '$\cdot$' stands for any possible action. In this model where the environment's actions are indifferent, it is clear that the nondeterministic strategy that allows both a and b in the initial position is a witness for x that satisfies φ_1'. We write this strategy $\sigma_{a,b}$, and we let σ_a and σ_b be the deterministic strategies that allow respectively only a and only b in the initial position (which is the only relevant part in this CGS).

If φ is a CTL^* formula, we now let

$$\varphi^{\mathrm{nd}}_{\mathrm{synth}}(\varphi) := \exists x.\forall y.(S,x)(E,y)\varphi$$

We can combine the two kinds of specifications (existential and universal) by requiring the existence of behaviors satisfying some properties, while requiring that all behaviors satisfy some other property. For instance, formula $\varphi^{\mathrm{nd}}_{\mathrm{synth}}(\mathbf{EF}p \wedge \mathbf{EF}q \wedge \mathbf{AGF}(p \vee q))$ asks that some outcomes reach p, some reach q, and all outcomes go infinitely often through p or q.

For arbitrary CTL^* specifications φ, formula $\varphi^{\mathrm{nd}}_{\mathrm{synth}}(\varphi)$ captures the *supervisory control problem* for CTL^* as defined in (Kupferman et al. 2000), where it is proved to be 3EXPTIME-complete. This formula has simulation depth at most 2, and thus the model-checking algorithm for $\mathsf{SL}^{\prec}$ has optimal complexity. The same complexity is proved in (Jiang and Kumar 2006) in a slightly different setting. If one changes $\exists x$ for $\exists^{\mathrm{d}}\, x$ in $\varphi^{\mathrm{nd}}_{\mathrm{synth}}(\varphi)$, one obtains instead the *synthesis problem with reactive environments*, also studied in (Kupferman et al. 2000), which has the same complexity.

4.3 Unilateral Forcing

We want to clarify what the synthesis problem with nondeterministic strategies as defined by $\varphi^{\mathrm{nd}}_{\mathrm{synth}}(\varphi)$ means when φ involves possibility requirements of the form $\mathbf{E}\psi$. Take for instance formula $\varphi^{\mathrm{nd}}_{\mathrm{synth}}(\mathbf{EG}p \wedge \mathbf{EF}\neg p)$, and consider the CGS $\mathcal{G}_2$ depicted in Figure 2. It is the case that $\mathcal{G}_2 \models \varphi^{\mathrm{nd}}_{\mathrm{synth}}(\mathbf{EG}p \wedge \mathbf{EF}\neg p)$, as witnessed by the strategy $\sigma_{a,b}$ that allows both a and b in the initial state. Indeed, if the system follows $\sigma_{a,b}$, then for any strategy σ' for the environment, there exists an outcome that satisfies $\mathbf{G}p$ and one that satisfies $\mathbf{F}\neg p$. In case strategy σ' is deterministic, the system can choose which kind of outcome to obtain: if σ' plays a in the initial position, and the system knows it, then the system can enforce $\mathbf{G}p$ by playing a in the initial position, which is allowed by its strategy σ, or it can choose to enforce $\mathbf{F}\neg p$ by playing b, also allowed by σ. If however the environment uses the nondeterministic strategy $\sigma_{a,b}$, then the system cannot unilaterally choose which possibility to enforce.

However, the ability to choose unilaterally can also be expressed in $\mathsf{SL}^{\prec}$, once more using the refinement operator. Define the following formula, where ψ is an LTL formula:

$$\mathrm{Force}(x,\psi) := \exists^{\mathrm{d}}\, y.\, y \preceq x \wedge (S,y)\mathbf{A}\psi$$

Now consider the formula

$$\varphi_2 := \exists x(S,x)\forall y(E,y)(\mathrm{Force}(x,\mathbf{G}p) \wedge \mathrm{Force}(x,\mathbf{F}\neg p))$$

It is easy to see that φ_2 implies $\varphi^{\mathrm{nd}}_{\mathrm{synth}}(\mathbf{EG}p \wedge \mathbf{EF}\neg p)$, but the converse is not true. When φ_2 holds it means, in addition to the existence of the two types of outcomes, that the system can unilaterally choose which of $\mathbf{G}p$ or $\mathbf{F}\neg p$ should hold, by picking one deterministic refinement of its nondeterministic strategy. For instance φ_2 holds on $\mathcal{G}_1$: if the system uses $\sigma_{a,b}$, it can choose to enforce $\mathbf{G}p$ by refining it to σ_a, or enforce $\mathbf{F}\neg p$ by refining it to σ_b. In both cases, the property will hold no matter how the environment behaves. However, φ_2 does not hold on $\mathcal{G}_2$: assume first that the system uses one of the deterministic strategies σ_a and σ_b; if the environment also picks a deterministic strategy, it fixes a unique outcome and thus $\mathrm{Force}(x,\mathbf{G}p) \wedge \mathrm{Force}(x,\mathbf{F}\neg p)$ does not hold. Now assume the system uses $\sigma_{a,b}$. If the environment also picks $\sigma_{a,b}$, then $\mathrm{Force}(x,\mathbf{G}p) \wedge \mathrm{Force}(x,\mathbf{F}\neg p)$ does not hold: indeed, the only deterministic refinements of $\sigma_{a,b}$ are σ_a and σ_b, and for any of these, the outcome ($\mathbf{G}p$ or $\mathbf{F}\neg p$) depends on which action the environment picks from its nondeterministic strategy $\sigma_{a,b}$.

This shows that φ_2 is indeed a stronger requirement than φ_1, which holds on both $\mathcal{G}_1$ and $\mathcal{G}_2$.

So in synthesis with nondeterministic strategies, we can require strategies that allow for different types of behaviors, and we have seen that we can also require that the system be able to choose which kind of behavior to enforce by refining its strategy. It is then natural to look for strategies that allow for as many behaviors as possible while satisfying the specification. Such strategies are usually called *maximally permissive* in the literature.

4.4 Maximally Permissive Synthesis

Different definitions of maximally permissive strategies exist. In supervisory control theory (Ramadge and Wonham 1987), maximality is expressed in terms of inclusion of sets of behaviors/outcomes, or equivalently by referring to simulation between the unfoldings of the systems where unauthorized transitions have been pruned, as in (Pinchinat and Riedweg 2005). In (Bernet, Janin, and Walukiewicz 2002), strategies are also compared by looking at inclusion of the behaviors/outcomes they allow. However, as it is proved in (Bernet, Janin, and Walukiewicz 2002), the existence of maximally permissive strategies for this notion of maximality is ensured only for simple safety games.

For this reason an alternative notion of strategy permissiveness was introduced in (Bouyer et al. 2009) for reachability games and further studied in (Bouyer et al. 2011) for parity games. In this setting, to each transition in a game is attached a cost that represents the penalty incurred by a strategy that does not allow this transition, and a maximally permissive strategy is one that minimizes penalties.

The latter definition ensures that maximally permissive strategies always exist, but the quantitative aspects involved, which are close to mean-payoff games (Gurvich, Karzanov, and Khachivan 1988), are known to quickly lead to undecidability when introduced in Strategy Logic (Gardy 2017). As for the definition based on inclusion of outcomes, it makes sense in the two-player antagonistic setting; but it is not adapted to our multi-player setting, where the set of outcomes induced by a strategy depends on which agent uses it, and which other agents have a defined strategy.

For this reason we consider the following natural definition of permissiveness based on refinement of strategies:

Definition 5. Strategy σ' is *more permissive than* strategy σ if $\sigma \prec \sigma'$.

Given a formula $\varphi(x)$, we can now express that a strategy x is maximally permissive with regards to $\varphi(x)$, i.e., that it satisfies $\varphi(x)$ and that no more permissive strategy satisfies it. Define formula $\mathrm{MaxPerm}(x, \varphi)$ as follows:

$$\mathrm{MaxPerm}(x, \varphi) \quad := \quad \varphi(x) \wedge (\forall y\ x \prec y \rightarrow \neg\varphi(y))$$

For instance, coming back to the framework of reactive synthesis, if the specification is a CTL^* formula φ, it holds that $\mathcal{G}, \chi, v_\iota \models \mathrm{MaxPerm}(x, \forall z(S, x)(E, z)\varphi)$ if, and only if, $\chi(x)$ is a maximally permissive system for specification φ.

Maximally permissive synthesis for a specification $\varphi \in \mathsf{CTL}^*$ can thus by expressed by the following $\mathsf{SL}^{\prec}$ formula:

$$\varphi^{\max}_{\text{synth}}(\varphi) := \exists x(a, x)\mathrm{MaxPerm}(x, \forall z(S, x)(E, z)\varphi)$$

When the formula is true, our model-checking algorithm can also produce a maximally permissive witness strategy for x. If φ is a CTL^* formula, $\varphi^{\max}_{\text{synth}}$ has simulation depth at most 3, and thus we obtain a 4EXPTIME upper-bound for this problem. We do not have the lower bounds, but we conjecture that one cannot do better, as the problem is already 3EXPTIME-complete without the constraint of maximal permissiveness (see Section 4.2), which seems to add one exponential to the complexity. However it can be reduced to 3EXPTIME for LTL specifications, i.e., when $\varphi = \mathbf{A}\psi$ with $\psi \in \mathsf{LTL}$ (we omit details for lack of space).

4.5 Strategy Refinement

One problem that is of interest in AI is that of producing plans that enforce some safety property, and in addition can at any time be refined to reach some secondary goal (Wright, Mattmüller, and Nebel 2018; Gerevini and Percassi 2019). For instance, consider an electric vehicle transporting rocks from a point A, where they are cut, to a point B, where they are used. The truck must ensure that the stock of rocks at point B never runs out (this is a safety property). We would like to synthesize a strategy for the truck such that this property is satisfied, but also so that at any time, the strategy can be refined to make the truck go through point C, where its battery can be reloaded. The fact that the strategy to reload refines the initial one ensures that the main property remains satisfied, i.e., it remains true that B will not run out even when the truck decides to go and reload. We can express this problem in $\mathsf{SL}^{\prec}$ as follows, where "empty" holds when there are no more rocks in point B, and "reload" means that the truck is at point C, reloading its battery.

$$\varphi_{\text{ref}} := \exists x(\text{truck}, x)\mathbf{AG}\big(\neg\text{empty}\wedge (\exists y.y \preceq x \wedge (\text{truck}, y)\mathbf{AF}\text{reload})\big)$$

The simulation depth for φ_{ref} is 2, and by Theorem 2 we obtain a 3EXPTIME upper bound for this problem, where $\mathbf{AF}$reload can be replaced by arbitrary CTL^* formulas. However for the particular formula φ_{ref} we can simplify the procedure to obtain a synthesis algorithm that runs in single exponential time (we omit details for lack of space).

4.6 Module Checking

Module checking (Kupferman, Vardi, and Wolper 2001; Jamroga and Murano 2014) is a generalisation of model checking to the setting of open systems, i.e., systems that interact with an environment. In this problem the system's nondeterministic strategy is fixed, and fixing a nondeterministic strategy for the environment thus yields a computation tree which is a pruning of the full system's computation tree. The problem then consists in checking that a property, specified for instance in CTL^*, holds in all such computation trees. The module-checking problem for a CTL^* formula φ can be written as follows in $\mathsf{SL}^{\prec}$:

$$\varphi_{\text{mod}} := \forall y(E, y)\,\varphi$$

Solving the module-checking problem for CTL^* specifications can thus be done by model checking φ_{mod}. This formula has simulation depth 1, and the procedure thus runs in doubly exponential time, which is asymptotically optimal (Kupferman, Vardi, and Wolper 2001).

4.7 Multi-agent Setting

One advantage of Strategy Logic is that it can naturally express complex game-theoretic solution concepts such as Nash equilibria or subgame-perfect equilibria. Since in $\mathsf{SL}^{\prec}$ we can express that a strategy profile (i.e., a strategy for each agent) refines another, we can also reason about refinements of such equilibria, and even synthesise maximally permissive equilibria. Assume that $\mathrm{Ag} = \{a_i : i \in [1,n]\}$, each agent a_i has objective $\psi_i \in \mathsf{LTL}$, and $(\boldsymbol{a}, \boldsymbol{x})$ stands for $(a_1, x_1)\dots(a_n, x_n)$ where $\boldsymbol{a} = (a_i)_{i\in[1,n]}$ and $\boldsymbol{x} = (x_i)_{i\in[1,n]}$. Consider for instance the formula

$$\varphi_{\mathrm{NE}}(\boldsymbol{x}) := (\boldsymbol{a}, \boldsymbol{x}) \bigwedge_{i\in[n]} \Big[\Big(\exists^{\mathrm{d}}\, y_i(a_i, y_i)\, \mathbf{A}\psi_i\Big) \rightarrow \mathbf{A}\psi_i\Big]$$

Formula φ_{NE} holds with assignment χ if, and only if, $(\chi(x_i))_{i\in[1,n]}$ is a Nash equilibrium. A profile $\boldsymbol{y}$ is more permissive than $\boldsymbol{x}$ if each y_i is at least as permissive as x_i, and and at least one refinement is strict. We thus let

$$\boldsymbol{x} \prec \boldsymbol{y} := \wedge_{i\in[1,n]} x_i \preceq y_i \wedge \vee_{i\in[1,n]} x_i \prec y_i$$

Now generalise formula $\mathrm{MaxPerm}(x, \varphi)$ from Section 4.4 as follows:

$$\mathrm{MaxPerm}^n(\boldsymbol{x}, \varphi) \quad := \quad \varphi(\boldsymbol{x}) \wedge (\forall^{\mathrm{d}}\, \boldsymbol{y}.\ \boldsymbol{x} \prec \boldsymbol{y} \rightarrow \neg\varphi(\boldsymbol{y}))$$

Synthesis of a maximally permissive Nash equilibrium can now be expressed with the following formula:

$$\exists^{\mathrm{d}}\, \boldsymbol{x}\, \mathrm{MaxPerm}^n(\boldsymbol{x}, \varphi_{\mathrm{NE}}(\boldsymbol{x}))$$

This formula having simulation depth 3, we obtain a 4EXPTIME upper bound on the complexity of this problem.

We can also generalise the notion of unilateral forcing from Section 4.3 by requiring that a subset of agents can together enforce a possibility by refining their strategies.

5 Model Checking $\mathsf{SL}^{\prec}$

We now turn to establishing that the model-checking problem for $\mathsf{SL}^{\prec}$ is decidable. To do so we extend the classic approach, which is to reduce to QCTL^*, the extension of CTL^* with quantification on atomic propositions. This logic is equivalent to MSO on infinite trees (Laroussinie and Markey 2014), and it is easy to express that a strategy (or the atomic propositions that code for it) refines another one.

Definition 6. The syntax of QCTL^* is defined as follows:

$$\varphi := p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{E}\psi \mid \exists p\, \varphi$$
$$\psi := \varphi \mid \neg\psi \mid \psi \vee \psi \mid \mathbf{X}\psi \mid \psi\mathbf{U}\psi$$

where $p \in \mathrm{AP}$.

Again, formulas of type φ are called *state formulas*, those of type ψ are called *path formulas*, and QCTL^* consists of all the state formulas defined by the grammar, and we use standard abbreviation $\mathbf{A}\psi := \neg\mathbf{E}\neg\psi$.

The models of QCTL^* are classic Kripke structures:

Definition 7. A *Kripke structure*, or KS, over AP is a tuple $\mathcal{S} = (S, R, \ell, s_\iota)$ where S is a set of *states*, $R \subseteq S \times S$ is a left-total[1] *transition relation*, $\ell : S \to 2^{\mathrm{AP}}$ is a *labeling function* and $s_\iota \in S$ is an *initial state*.

[1] i.e., for all $s \in S$, there exists s' such that $(s, s') \in R$.

A *path* in $\mathcal{S}$ is an infinite sequence of states $\lambda = s_0 s_1 \dots$ such that for all $i \in \mathbb{N}$, $(s_i, s_{i+1}) \in R$. A *finite path* is a finite non-empty prefix of a path. Similar to continuations of finite plays, given a finite path λ we write $\mathrm{Cont}(\lambda)$ for the set of finite paths that start with λ. The *size* $|\mathcal{S}|$ of a KS $\mathcal{S} = (S, R, s_\iota, \ell)$ is its number of states: $|\mathcal{S}| := |S|$.

Since we will evaluate QCTL^* formulas on tree-unfoldings of Kripke structures, we need the following definitions for infinite trees.

Trees. Let X be a finite set of *directions* (typically a set of states). An *X-tree* τ is a nonempty set of words $\tau \subseteq X^+$ such that **(1)** there exists $r \in X$, called the *root* of τ, such that each $u \in \tau$ starts with r ($r \preccurlyeq u$); **(2)** if $u \cdot x \in \tau$ and $u \cdot x \neq r$, then $u \in \tau$; **(3)** if $u \in \tau$ then there exists $x \in X$ such that $u \cdot x \in \tau$.

The elements of a tree τ are called *nodes*. If $u \cdot x \in \tau$, we say that $u \cdot x$ is a *child* of u. A *path* in τ is an infinite sequence of nodes $\lambda = u_0 u_1 \dots$ such that for all $i \in \mathbb{N}$, u_{i+1} is a child of u_i, and $Paths(u)$ is the set of paths that start in node u.

An AP-*labeled X-tree*, or (AP, X)-*tree* for short, is a pair $t = (\tau, \ell)$, where τ is an X-tree called the *domain* of t and $\ell : \tau \to 2^{\mathrm{AP}}$ is a *labeling*, which maps each node to the set of propositions that hold in it. For $p \in \mathrm{AP}$, a *p-labeling* for a tree is a mapping $\ell_p : \tau \to \{0, 1\}$ that indicates in which nodes p holds, and for a labeled tree $t = (\tau, \ell)$, the p-labeling of t is the p-labeling $u \mapsto 1$ if $p \in \ell(u)$, 0 otherwise. The composition of a labeled tree $t = (\tau, \ell)$ with a p-labeling ℓ_p for τ is defined as $t \otimes \ell_p := (\tau, \ell')$, where $\ell'(u) = \ell(u) \cup \{p\}$ if $\ell_p(u) = 1$, and $\ell(u) \setminus \{p\}$ otherwise. A p-labeling for a labeled tree $t = (\tau, \ell)$ is a p-labeling for its domain τ. A *pointed labeled tree* is a pair (t, u) where u is a node of t.

Let $\mathcal{S} = (S, R, \ell, s_\iota)$ be a Kripke structure over AP. The *tree-unfolding of $\mathcal{S}$* is the (AP, S)-tree $t_{\mathcal{S}} := (\tau, \ell')$, where τ is the set of all finite paths that start in s_ι, and for every $u \in \tau$, $\ell'(u) := \ell(\mathrm{last}(u))$.

Definition 8. We define by induction the satisfaction relation $\models$ of QCTL^*. Let $t = (\tau, \ell)$ be an AP-labeled tree, u a node and λ a path in τ (we omit Boolean cases):

$t, u \models p$	if	$p \in \ell(u)$
$t, u \models \mathbf{E}\psi$	if	$\exists\, \lambda \in Paths(u)$ s.t. $t, \lambda \models \psi$
$t, u \models \exists p\, \varphi$	if	$\exists\, \ell_p$ a p-labeling for t s.t. $t \otimes \ell_p, u \models \varphi$
$t, \lambda \models \varphi$	if	$t, \lambda_0 \models \varphi$
$t, \lambda \models \mathbf{X}\psi$	if	$t, \lambda_{\geq 1} \models \psi$
$t, \lambda \models \psi\mathbf{U}\psi'$	if	$\exists\, i \geq 0$ s.t. $t, \lambda_{\geq i} \models \psi'$ and $\forall j$ s.t. $0 \leq j < i,\ t, \lambda_{\geq j} \models \psi$

We write $t \models \varphi$ for $t, r \models \varphi$, where r is the root of t. Given a KS $\mathcal{S}$ and a QCTL^* formula φ, we write $\mathcal{S} \models \varphi$ if $t_{\mathcal{S}} \models \varphi$. The simulation depth for QCTL^* is defined exactly as for $\mathsf{SL}^{\prec}$, with the case for $\exists p\, \varphi$ corresponding to $\exists x\, \varphi$. The following is proved in (Laroussinie and Markey 2014):

Theorem 4. *Model checking QCTL^* is $(k+1)$-EXPTIME-complete for formulas of simulation depth at most k.*

5.1 Reduction to QCTL*

We use a variant of the reductions presented in (Laroussinie and Markey 2015; Fijalkow et al. 2018; Berthon et al. 2017; Maubert and Murano 2018; Bouyer et al. 2019), which transform instances of the model-checking problem for various strategic logics to (extensions of) QCTL*.

Let $(\mathcal{G}, \Phi)$ be an instance of the SL model-checking problem, and assume without loss of generality that each strategy variable is quantified at most once in Φ. We define an equivalent instance of the model-checking problem for QCTL*.

Define the KS $\mathcal{S}_\mathcal{G} := (S, R, s_\iota, \ell')$ where

- $S := \{s_v \mid v \in V\}$,
- $R := \{(s_v, s_{v'}) \mid \exists \boldsymbol{c} \in \mathrm{Ac}^{\mathrm{Ag}} \text{ s.t. } E(v, \boldsymbol{c}) = v'\} \subseteq S^2$,
- $s_\iota := s_{v_\iota}$, and
- $\ell'(s_v) := \ell(v) \cup \{p_v\} \subseteq \mathrm{AP} \cup \mathrm{AP}_v$.

For every finite play $\rho = v_0 \dots v_k$, define the node $u_\rho := s_{v_0} \dots s_{v_k}$ in $t_{\mathcal{S}_\mathcal{G}}$. The mapping $\rho \mapsto u_\rho$ defines a bijection between the set of finite plays and the set of nodes in $t_{\mathcal{S}_\mathcal{G}}$.

We now describe how to transform an $\mathsf{SL}^{\prec}$ formula φ and a partial function $f : \mathrm{Ag} \rightharpoonup \mathrm{Var}$ into a QCTL* formula $(\varphi)^f_s$ (that will also depend on $\mathcal{G}$). Suppose that $\mathrm{Ac} = \{c_1, \dots, c_l\}$, and define $(\varphi)^f_s$ and $(\psi)^f_p$ by mutual induction on state and path formulas. The base cases are as follows: $(p)^f_s := p$ and $(\varphi)^f_p := (\varphi)^f_s$. Boolean and temporal operators are simply obtained by distributing the translation: $(\neg\varphi)^f_s := \neg(\varphi)^f_s$, $(\neg\psi)^f_p := \neg(\psi)^f_p$, $(\varphi_1 \vee \varphi_2)^f_s := (\varphi_1)^f_s \vee (\varphi_2)^f_s$, $(\psi_1 \vee \psi_2)^f_p := (\psi_1)^f_p \vee (\psi_2)^f_p$, $(\mathbf{X}\psi)^f_p := \mathbf{X}(\psi)^f_p$ and $(\psi_1 \mathbf{U} \psi_2)^f_p := (\psi_1)^f_p \mathbf{U} (\psi_2)^f_p$.

We continue with the strategy quantifier:

$$(\exists x\, \varphi)^f_s := \exists p^x_{c_1} \dots \exists p^x_{c_l}. \varphi_{\mathrm{str}}(x) \wedge (\varphi)^f_s$$

where $\varphi_{\mathrm{str}}(x) := \mathbf{AG} \bigvee_{c \in \mathrm{Ac}} p^x_c$ and

$$(\exists^{\mathrm{d}} x\, \varphi)^f_s := \exists p^x_{c_1} \dots \exists p^x_{c_l}. \varphi^{\mathrm{det}}_{\mathrm{str}}(x) \wedge (\varphi)^f_s$$

where $\varphi^{\mathrm{det}}_{\mathrm{str}}(x) := \mathbf{AG} \bigvee_{c \in \mathrm{Ac}} (p^x_c \wedge \bigwedge_{c' \neq c} \neg p^x_{c'})$.

The intuition is that for each possible action $c \in \mathrm{Ac}$, an existential quantification on the atomic proposition p^x_c "chooses" for each node u_ρ of the tree $t_{\mathcal{S}_\mathcal{G}}$ whether strategy x allows action c in ρ or not. $\varphi_{\mathrm{str}}(x)$ checks that at least one action is allowed in each node, and thus that atomic propositions p^x_c indeed define a (nondeterministic) strategy. Formula $\varphi^{\mathrm{det}}_{\mathrm{str}}(x)$ instead ensures that *exactly one* action is chosen for strategy x in each finite play, and thus that atomic propositions p^x_c characterise a deterministic strategy.

For strategy refinement, the translation is as follows:

$$(x \preceq y)^f_s := \mathbf{AG} \bigwedge_{c \in \mathrm{Ac}} p^x_c \to p^y_c.$$

Here are the remaining cases:

$$((a, x)\varphi)^f_s := (\varphi)^{f[a \mapsto x]}_s \quad \text{for } x \in \mathrm{Var}$$

and $(\mathbf{E}\psi)^f_s := \mathbf{E}\,(\psi^f_{\mathrm{out}} \wedge (\psi)^f_p)$, where

$$\psi^f_{\mathrm{out}} := \mathbf{G} \bigvee_{v \in V} \Big(p_v \wedge \bigvee_{\boldsymbol{c} \in \mathrm{Ac}^{\mathrm{Ag}}} \big(\bigwedge_{a \in \mathit{dom}(f)} p^{f(a)}_{\boldsymbol{c}_a} \wedge \mathbf{X}\, p_{E(v,\boldsymbol{c})} \big) \Big).$$

Formula ψ^f_{out} checks that each player a in the domain of f follows the strategy coded by the $p^{f(a)}_c$.

To prove the correctness of the translation we need some additional definitions. First, given a strategy σ and a strategy variable x we let $\ell^x_\sigma := \{\ell_{p^x_c} \mid c \in \mathrm{Ac}\}$ be the family of p^x_c-labelings for tree $t_{\mathcal{S}_\mathcal{G}}$ defined as follows: for each finite play ρ and $c \in \mathrm{Ac}$, we let $\ell_{p^x_c}(u_\rho) := 1$ if $c \in \sigma(\rho)$, 0 otherwise. For a labeled tree t with same domain as $t_{\mathcal{S}_\mathcal{G}}$ we write $t \otimes \ell^x_\sigma$ for $t \otimes \ell_{p^x_{c_1}} \otimes \dots \otimes \ell_{p^x_{c_l}}$.

Second, given an infinite play π and a point $i \in \mathbb{N}$, we let $\lambda_{\pi,i}$ be the infinite path in $t_{\mathcal{S}_\mathcal{G}}$ that starts in node $u_{\pi_{\leq i}}$ and is defined as $\lambda_{\pi,i} := u_{\pi_{\leq i}} u_{\pi_{\leq i+1}} u_{\pi_{\leq i+2}} \dots$

Finally, we say that a partial function $f : \mathrm{Ag} \rightharpoonup \mathrm{Var}$ is *compatible* with an assignment χ if $\mathit{dom}(\chi) \cap \mathrm{Ag} = \mathit{dom}(f)$ and for all $a \in \mathit{dom}(f)$, $\chi(a) = \chi(f(a))$.

Proposition 5. *For every state subformula φ and path subformula ψ of Φ, finite play ρ, infinite play π, point $i \in \mathbb{N}$, for every assignment χ variable-complete for φ (resp. ψ) and partial function $f : \mathrm{Ag} \rightharpoonup \mathrm{Var}$ compatible with χ, assuming also that no x_i in $\mathit{dom}(\chi) \cap \mathrm{Var} = \{x_1, \dots, x_k\}$ is quantified in φ or ψ, we have*

$$\mathcal{G}, \chi, \rho \models \varphi \quad \textit{iff} \quad t_{\mathcal{S}_\mathcal{G}} \otimes \ell^{x_1}_{\chi(x_1)} \otimes \dots \ell^{x_k}_{\chi(x_k)}, u_\rho \models (\varphi)^f_s$$

$$\mathcal{G}, \chi, \pi, i \models \psi \quad \textit{iff} \quad t_{\mathcal{S}_\mathcal{G}} \otimes \ell^{x_1}_{\chi(x_1)} \otimes \dots \ell^{x_k}_{\chi(x_k)}, \lambda_{\pi,i} \models (\psi)^f_p$$

In addition, $\mathcal{S}_\mathcal{G}$ is of size linear in $|\mathcal{G}|$, and $(\varphi)^f_s$ and $(\psi)^f_p$ are of size linear in $|\mathcal{G}|^2 + |\varphi|$.

Proof. The proof is by induction on φ. We detail the case for binding, strategy quantification, strategy refinement and outcome quantification, the others follow simply by definition of $\mathcal{S}_\mathcal{G}$ for atomic propositions and induction hypothesis for remaining cases.

For $\varphi = x \preceq y$, assume that $\mathcal{G}, \chi, \rho \models x \preceq y$. First, observe that since χ is variable-complete for φ, x and y are in $\mathit{dom}(\chi)$. Now we have that $\chi(x)_{|\rho}(\rho') \subseteq \chi(y)_{|\rho}(\rho')$ for every $\rho' \in \mathrm{Cont}(\rho)$. By definition of $\ell^x_{\chi(x)} = \{\ell_{p^x_c} \mid c \in \mathrm{Ac}\}$ and $\ell^y_{\chi(y)} = \{\ell_{p^y_c} \mid c \in \mathrm{Ac}\}$, it follows that for each $c \in \mathrm{Ac}$ and $\rho' \in \mathrm{Cont}(\rho)$, if $\ell_{p^x_c}(\rho') = 1$, then $\ell_{p^y_c}(\rho') = 1$, and thus

$$t_{\mathcal{S}_\mathcal{G}} \otimes \ell^x_{\chi(x)} \otimes \ell^y_{\chi(y)} \models \mathbf{AG} \bigwedge_{c \in \mathrm{Ac}} p^x_c \to p^y_c$$

The result then holds since the labellings $\ell^x_{\chi(x)}$ touch distinct sets of atomic propositions for each variable x in $\mathrm{Var}(\chi)$.

For the other direction let $t = t_{\mathcal{S}_\mathcal{G}} \otimes \ell^{x_1}_{\chi(x_1)} \otimes \dots \otimes \ell^{x_k}_{\chi(x_k)}$ and assume that

$$t, u_\rho \models \mathbf{AG} \bigwedge_{c \in \mathrm{Ac}} p^x_c \to p^y_c.$$

This implies that for every $\rho' \in \mathrm{Cont}(\rho)$,

$$t, u_{\rho'} \models \bigwedge_{c \in \mathrm{Ac}} p^x_c \to p^y_c,$$

and thus $\chi(x)_{|\rho}$ refines $\chi(y)_{|\rho}$.

For $\varphi = (a, x)\varphi'$, we have $\mathcal{G}, \chi, \rho \models (a, x)\varphi'$ if and only if $\mathcal{G}, \chi[a \mapsto \chi(x)], \rho \models \varphi'$. The result follows by using the

induction hypothesis with assignment $\chi[a \mapsto x]$ and function $f[a \mapsto x]$. This is possible because $f[a \mapsto x]$ is compatible with $\chi[a \mapsto x]$: indeed $\mathit{dom}(\chi[a \mapsto x]) \cap \mathrm{Ag}$ is equal to $\mathit{dom}(\chi) \cap \mathrm{Ag} \cup \{a\}$ which, by assumption, is equal to $\mathit{dom}(f) \cup \{a\} = \mathit{dom}(f[a \mapsto x])$. Also by assumption, for all $a' \in \mathit{dom}(f)$, $\chi(a') = \chi(f(a'))$, and by definition $\chi[a \mapsto \chi(x)](a) = \chi(x) = \chi(f[a \mapsto x](a))$.

For $\varphi = \exists x \varphi'$, assume first that $\mathcal{G}, \chi, \rho \models \exists x \varphi'$. There exists a nondeterministic strategy σ such that

$$\mathcal{G}, \chi[x \mapsto \sigma], \rho \models \varphi'.$$

Since f is compatible with χ, it is also compatible with assignment $\chi' = \chi[x \mapsto \sigma]$. By assumption, no variable in $\{x_1, \dots, x_k\}$ is quantified in φ, so that $x \neq x_i$ for all i, and thus $\chi'(x_i) = \chi(x_i)$ for all i; and because no strategy variable is quantified twice in a same formula, x is not quantified in φ', so that no variable in $\{x_1, \dots, x_k, x\}$ is quantified in φ'. By induction hypothesis

$$t_{\mathcal{S}_\mathcal{G}} \otimes \ell^{x_1}_{\chi'(x_1)} \otimes \dots \otimes \ell^{x_k}_{\chi'(x_k)} \otimes \ell^{x}_{\chi'(x)}, u_\rho \models (\varphi')^f_s.$$

It follows that

$$t_{\mathcal{S}_\mathcal{G}} \otimes \ell^{x_1}_{\chi'(x_1)} \otimes \dots \otimes \ell^{x_k}_{\chi'(x_k)}, u_\rho \models \exists p^x_{c_1} \dots p^x_{c_l}.\varphi_{\mathrm{str}}(x) \wedge (\varphi')^f_s$$

Finally, since $\chi'(x_i) = \chi(x_i)$ for all i, we conclude that

$$t_{\mathcal{S}_\mathcal{G}} \otimes \ell^{x_1}_{\chi(x_1)} \otimes \dots \otimes \ell^{x_k}_{\chi(x_k)}, u_\rho \models (\exists x \varphi')^f_s.$$

For the other direction, assume that

$$t_{\mathcal{S}_\mathcal{G}} \otimes \ell^{x_1}_{\chi(x_1)} \otimes \dots \otimes \ell^{x_k}_{\chi(x_k)}, u_\rho \models (\varphi)^f_s,$$

and recall that $(\varphi)^f_s = \exists p^x_{c_1} \dots \exists p^x_{c_l}.\varphi_{\mathrm{str}}(x) \wedge (\varphi')^f_s$. Write $t = t_{\mathcal{S}_\mathcal{G}} \otimes \ell^{x_1}_{\chi(x_1)} \otimes \dots \otimes \ell^{x_k}_{\chi(x_k)}$. There exist $\ell_{p^x_c}$-labelings such that

$$t \otimes \ell_{p^x_{c_1}} \otimes \dots \otimes \ell_{p^x_{c_l}} \models \varphi_{\mathrm{str}}(x) \wedge (\varphi')^f_s.$$

By $\varphi_{\mathrm{str}}(x)$, these labelings code for a strategy σ. Let $\chi' = \chi[x \mapsto \sigma]$. For all $1 \leq i \leq k$, by assumption $x \neq x_i$, and thus $\chi'(x_i) = \chi(x_i)$. The above can thus be rewritten

$$t_{\mathcal{S}_\mathcal{G}} \otimes \ell^{x_1}_{\chi'(x_1)} \otimes \dots \otimes \ell^{x_k}_{\chi'(x_k)} \otimes \ell^{x}_{\chi'(x)} \models \varphi_{\mathrm{str}}(x) \wedge (\varphi')^f_s.$$

By induction hypothesis we have $\mathcal{G}, \chi[x \mapsto \sigma], \rho \models \varphi'$, hence $\mathcal{G}, \chi, \rho \models \exists x \varphi'$.

For $\varphi = \exists^{\mathrm{d}} x \varphi$ the proof is similar, using $\varphi^{\mathrm{det}}_{\mathrm{str}}(x)$ instead of $\varphi_{\mathrm{str}}(x)$.

For $\varphi = \mathbf{E}\psi$, assume that $\mathcal{G}, \chi, \rho \models \mathbf{E}\psi$. There exists a play $\pi \in \mathrm{Out}(\chi, \rho)$ s.t. $\mathcal{G}, \chi, \pi, |\rho| - 1 \models \psi$. By induction hypothesis,

$$t_{\mathcal{S}_\mathcal{G}} \otimes \ell^{x_1}_{\chi(x_1)} \otimes \dots \otimes \ell^{x_k}_{\chi(x_k)}, \lambda_{\pi,|\rho|-1} \models (\psi)^f_p.$$

Since π is an outcome of χ, each agent $a \in \mathit{dom}(\chi) \cap \mathrm{Ag}$ follows strategy $\chi(a)$ in π. Because $\mathit{dom}(\chi) \cap \mathrm{Ag} = \mathit{dom}(f)$ and for all $a \in \mathit{dom}(f)$, $\chi(a) = \chi(f(a))$, each agent $a \in \mathit{dom}(f)$ follows the strategy $\chi(f(a))$, which is coded by atoms $p^{f(a)}_c$ in the translation of Φ. Therefore $\lambda_{\pi,|\rho|-1}$ also satisfies ψ^χ_{out}, hence

$$t_{\mathcal{S}_\mathcal{G}} \otimes \ell^{x_1}_{\chi(x_1)} \otimes \dots \otimes \ell^{x_k}_{\chi(x_k)}, \lambda_{\pi,|\rho|-1} \models \psi^\chi_{\mathrm{out}} \wedge (\psi)^f_p.$$

For the other direction, assume that

$$t_{\mathcal{S}_\mathcal{G}} \otimes \ell^{x_1}_{\chi(x_1)} \otimes \dots \otimes \ell^{x_k}_{\chi(x_k)}, u_\rho \models \mathbf{E}(\psi^f_{\mathrm{out}} \wedge (\psi)^f_p).$$

There exists a path λ in $t_{\mathcal{S}_\mathcal{G}} \otimes \ell^{x_1}_{\chi(x_1)} \otimes \dots \otimes \ell^{x_k}_{\chi(x_k)}$ starting in node u_ρ that satisfies both ψ^f_{out} and $(\psi)^f_p$. By construction of $\mathcal{S}_\mathcal{G}$ there exists an infinite play π such that $\pi_{\leq|\rho|-1} = \rho$ and $\lambda = \lambda_{\pi,|\rho|-1}$. By induction hypothesis, $\mathcal{G}, \chi, \pi, |\rho|-1 \models \psi$. Because $\lambda_{\pi,|\rho|-1}$ satisfies ψ^f_{out}, $\mathit{dom}(\chi) \cap \mathrm{Ag} = \mathit{dom}(f)$, and for all $a \in \mathit{dom}(f)$, $\chi(a) = \chi(f(a))$, it is also the case that $\pi \in \mathrm{Out}(\chi, \rho)$, hence $\mathcal{G}, \chi, \rho \models \mathbf{E}\psi$. □

Applying Proposition 5 to the sentence Φ, $\rho = v_\iota$, any assignment χ, and the empty function $\emptyset$, we get:

$$\mathcal{G} \models \Phi \quad \text{if and only if} \quad t_{\mathcal{S}_\mathcal{G}} \models (\Phi)^\emptyset_s.$$

To obtain the upper bounds of Theorem 2, we use the above equivalence, Theorem 4, and the fact that the translation $()^f_s$ preserves simulation depth, modulo the following details: We have $(\exists x\, \varphi)^f_s = \exists p^x_{c_1} \dots \exists p^x_{c_l}.\varphi_{\mathrm{str}}(x) \wedge (\varphi)^f_s$, and because of the conjunction $\mathrm{sd}_x(\varphi_{\mathrm{str}}(x) \wedge (\varphi)^f_s) = \mathrm{alt}$. It follows that when $\mathrm{sd}_x(\varphi) = \mathrm{nd}$, $\mathrm{sd}_k((\exists x\, \varphi)^f_s)$ is one more than $\mathrm{sd}_k(\exists x\, \varphi)$. But in this precise case conjunction does not introduce alternation, and if the automaton for $(\varphi)^f_s$ is nondeterministic we can obtain a nondeterministic automaton for $\varphi_{\mathrm{str}}(x) \wedge (\varphi)^f_s$ without incurring an exponential blowup. The reason is that $\varphi_{\mathrm{str}}(x)$ can be recognized by a very simple *deterministic* tree automaton (see (Berthon et al. 2020, p.30) for details). The case for $\exists^{\mathrm{d}} x\, \varphi$ is similar. For $x \preceq y$ we have $(x \preceq y)_s = \mathbf{AG} \bigwedge_{c \in \mathrm{Ac}} p^x_c \to p^y_c$ hence $\mathrm{sd}_x((x \preceq y)_s) = \mathrm{alt}$, but $\mathrm{sd}_x(x \preceq y) = \mathrm{nd}$. Again, in this particular case, one can obtain a *deterministic* automaton with two states for $(x \preceq y)_s$, which solves the matter.

The lower bounds of Theorem 2 are inherited from those for SL without refining operator (Berthon et al. 2020).

6 Conclusion

In this work we extended Strategy Logic with nondeterministic strategies and a refinement operator that expresses that a strategy is more permissive than another. We showed how the resulting logic $\mathsf{SL}^\prec$ captures in a natural manner a variety of problems previously not expressible in Strategy Logic, such as module checking, synthesis with reactive environments or synthesis of maximally permissive strategies. We also showed how the refinement operator allows us to specify meaningful requirements for nondeterministic synthesis that are not expressible in CTL^*, such as unilateral forcing. We solved the model-checking problem for $\mathsf{SL}^\prec$ by reduction to QCTL^*, and we established its complexity in terms of the simulation depth of the formulas. This precise measure shows that, for the problems from the literature whose precise complexity is known, the synthesis procedures that we obtain via $\mathsf{SL}^\prec$ have optimal complexity. The model-checking algorithm for $\mathsf{SL}^\prec$ also provides synthesis procedures for problems that, up to our knowledge, have not been solved before, such as synthesis with unilateral forcing specifications or synthesis of maximally permissive Nash equilibria. As future work we plan to establish the precise complexity of these problems to see if our procedure is optimal.

References

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-time temporal logic. *J. ACM* 49(5):672–713.

Banihashemi, B.; De Giacomo, G.; and Lespérance, Y. 2018. Hierarchical agent supervision. In *AAMAS*.

Belardinelli, F.; Lomuscio, A.; Murano, A.; and Rubin, S. 2020. Verification of multi-agent systems with public actions against strategy logic. *Artif. Intell.* 103302.

Bernet, J.; Janin, D.; and Walukiewicz, I. 2002. Permissive strategies: from parity games to safety games. *RAIRO-Theoretical Informatics and Applications* 36(3):261–275.

Berthon, R.; Maubert, B.; Murano, A.; Rubin, S.; and Vardi, M. Y. 2017. Strategy logic with imperfect information. In *LICS*, 1–12. IEEE.

Berthon, R.; Maubert, B.; Murano, A.; Rubin, S.; and Vardi, M. 2020. Strategy logic with imperfect information. *arXiv* 2003.04730.

Bouyer, P.; Duflot, M.; Markey, N.; and Renault, G. 2009. Measuring permissivity in finite games. In *CONCUR*, 196–210. Springer.

Bouyer, P.; Markey, N.; Olschewski, J.; and Ummels, M. 2011. Measuring permissiveness in parity games: Mean-payoff parity games revisited. In *ATVA*, 135–149. Springer.

Bouyer, P.; Kupferman, O.; Markey, N.; Maubert, B.; Murano, A.; and Perelli, G. 2019. Reasoning about quality and fuzziness of strategic behaviours. In *IJCAI*, 1588–1594.

Čermák, P.; Lomuscio, A.; Mogavero, F.; and Murano, A. 2018. Practical verification of multi-agent systems against slk specifications. *Inf. Comput.* 261:588–614.

Chatterjee, K.; Henzinger, T. A.; and Piterman, N. 2010. Strategy logic. *Inf. Comput.* 208.

De Giacomo, G.; Lespérance, Y.; and Muise, C. J. 2012. On supervising agents in situation-determined ConGolog. In *AAMAS*, 1031–1038.

De Giacomo, G.; Patrizi, F.; and Sardiña, S. 2013. Automatic behavior composition synthesis. *Artif. Intell.* 196:106–142.

Fijalkow, N.; Maubert, B.; Murano, A.; and Rubin, S. 2018. Quantifying bounds in strategy logic. In *CSL*, 23:1–23:23.

Gardy, P. 2017. *Semantics of Strategy Logic*. Theses, Université Paris-Saclay.

Geffner, H., and Bonet, B. 2013. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers.

Gerevini, A. E., and Percassi, F. 2019. On compiling away PDDL3 soft trajectory constraints without using automata. In Benton, J.; Lipovetzky, N.; Onaindia, E.; Smith, D. E.; and Srivastava, S., eds., *ICAPS*, 320–328.

Gurvich, V. A.; Karzanov, A. V.; and Khachivan, L. 1988. Cyclic games and an algorithm to find minimax cycle means in directed graphs. *USSR Computational Mathematics and Mathematical Physics* 28(5):85–91.

Jamroga, W., and Murano, A. 2014. On module checking and strategies. In *AAMAS*, 701–708. International Foundation for Autonomous Agents and Multiagent Systems.

Jiang, S., and Kumar, R. 2006. Supervisory control of discrete event systems with ctl* temporal logic specifications. *SIAM. J. Control Optim.* 44(6):2079–2103.

Kupferman, O., and Vardi, M. Y. 1997. Module checking revisited. In *CAV*, 36–47. Springer.

Kupferman, O.; Madhusudan, P.; Thiagarajan, P.; and Vardi, M. Y. 2000. Open systems in reactive environments: Control and synthesis. In *CONCUR*, 92–107. Springer.

Kupferman, O.; Vardi, M. Y.; and Wolper, P. 2001. Module checking. *Inf. Comput.* 164(2):322–344.

Laroussinie, F., and Markey, N. 2014. Quantified CTL: expressiveness and complexity. *Log. Meth. Comput. Sci.* 10(4).

Laroussinie, F., and Markey, N. 2015. Augmenting ATL with strategy contexts. *Inf. Comput.* 245:98–123.

Maubert, B., and Murano, A. 2018. Reasoning about knowledge and strategies under hierarchical information. In *KR*, 530–540.

Mogavero, F.; Murano, A.; Perelli, G.; and Vardi, M. Y. 2014. Reasoning about strategies: On the model-checking problem. *ACM Trans. Comput. Log.* 15(4):34:1–34:47.

Pinchinat, S., and Riedweg, S. 2005. You can always compute maximally permissive controllers under partial observation when they exist. In *ACC*, 2287–2292. IEEE.

Pnueli, A., and Rosner, R. 1989. On the synthesis of a reactive module. In *POPL*, 179–190.

Ramadge, P. J., and Wonham, W. M. 1987. Supervisory control of a class of discrete event processes. *SIAM journal on control and optimization* 25(1):206–230.

Wonham, W. M. 2014. *Supervisory Control of Discrete-Event Systems*. University of Toronto, 2014 edition.

Wright, B.; Mattmüller, R.; and Nebel, B. 2018. Compiling away soft trajectory constraints in planning. In Thielscher, M.; Toni, F.; and Wolter, F., eds., *KR*, 474–483.