

Reasoning about Strategies

Fabio Mogavero¹ Aniello Murano¹ Moshe Y. Vardi²

¹Università degli Studi di Napoli "Federico II"

<http://people.na.infn.it/~{mogavero,murano}>

²Rice University

<http://www.cs.rice.edu/~vardi/>

30th IARCS Annual Conference on
Foundations of Software Technology and Theoretical Computer Science
Chennai, India, December 15-18, 2010

Aim of our work

Idea

We are looking for a logic in which we can talk about the **strategic behavior** of agents in **multi-player games**.

Application

It can be used as a specification language for the formal verification and synthesis of **modular and interactive systems**.

From monolithic to multi-agent systems

Historical development:

- **Model checking**: analyzes systems monolithically (system components plus environment) [Clarke & Emerson, Queille & Sifakis, '81].
- **Module checking**: separates out the environment from the system components, but still views the system monolithically [Kupferman & Vardi, '96].
- **Alternating temporal reasoning**: multi-agent systems (components individually considered), playing strategically [Alur et al., '02].
- **Strategic logic reasoning**: two-player turn-based games verified by considering strategies as first order objects [Chatterjee et al., '07].

Strategic reasoning

Example

Reactive synthesis: Synthesize an interactive system that satisfies a given specification, independently of the possible sequences of inputs.

Example

Nash equilibrium: Verify that the players of a game have optimal strategies (each player is willing to follow its optimal strategy, if also the other do that).

Concurrent game model (I)

A *concurrent game structure* is a tuple $\mathcal{G} = \langle AP, Ag, Ac, St, \lambda, \tau, s_0 \rangle$ where:

- 1 AP : set of *atomic propositions*;
- 2 Ag : set of *agents*;
- 3 Ac : set of *actions*;
- 4 St : set of *states*;
- 5 $s_0 \in St$: *designated initial state*;
- 6 $\lambda : St \rightarrow 2^{AP}$: *labeling function*;
- 7 $\tau : St \times Ac^{Ag} \rightarrow St$: *transition function* mapping a state and a *decision* (i.e., a function from Ag to Ac) to a new state.

Concurrent game model (II)

S_t is not the global state space of the system, but the state space of the environment (the game) in which the agents operate.

A_c consists of local actions of all the agents.

$A_c^{A_g}$ represents the set of choices of an action for each agent.

Alternating-Time Temporal Logics [Alur et al., '02]

$\langle\langle A \rangle\rangle\psi$: There is a strategy for the agents in $A \subseteq \text{Ag}$ enforcing the property ψ , independently of what the agents not in A can do.

Example

$\langle\langle \{\alpha, \beta\} \rangle\rangle G \neg \text{fail}$: “Agents α and β cooperate to ensure that a system (having possibly more than two processes (agents)) never enters a fail state.”

Despite its powerful expressiveness, ATL* suffers from the strong limitation that strategies are treated only implicitly.

The quantifier alternation is fixed to 1!

For instance, in the previous example, you cannot say that α has an uniform strategy w.r.t. the other agents, while β can choose its in dependence of their.

Strategy Logic [Chatterjee et al., '07]

Example

$\exists x_1, x_2. \forall y. \psi(x_1, y) \wedge \neg \psi(x_2, y)$: There are two strategies for the first player, x_1 and x_2 , such that x_1 enforces ψ while x_2 enforces $\neg \psi$, independently of the strategy y of the second player.

- The logic is investigated only under the weak framework of two-players and turn-based games.
- The model checking procedure is non-elementary, without matching lower-bound.
- The satisfiability problem is not studied at all.

Our contribution

We introduce a new **Strategy Logic** (SL), as a more general framework (both in its syntax and semantics), for explicit reasoning about strategies in *multi-player concurrent games*.

We investigate and solve both the problems of

- **model-checking** (decidable, **2EXPTIME-COMplete**),
- **satisfiability** (undecidable, **Σ_1^1 -HARD**).

Outline

- 1 Reasoning about Strategies
 - Syntax and Informal Semantics
 - Examples of Specifications
- 2 Decision problems
 - Model Checking
 - Satisfiability
- 3 Conclusion

Underlying temporal logic

LTL: Linear-Time Temporal Logic [Pnueli, '79]

$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi U\varphi \mid \varphi R\varphi.$

Example

- $X\varphi$: φ holds in the next state;
- $F\varphi$: φ holds eventually;
- $G\varphi$: φ holds forever;
- $\varphi_1 U \varphi_2$: φ_1 holds until φ_2 holds;
- $\varphi_1 R \varphi_2$: φ_2 holds forever or until φ_1 holds.

Syntax of SL

Definition

SL *formulas* are built inductively in the following way, where $p \in AP$ is an atomic proposition, $x \in Var$ a variable, and $a \in Ag$ an agent.

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

SL syntactically extends LTL by means of *strategy quantifiers*, the existential $\langle\langle x \rangle\rangle$ and the universal $[\![x]\!]$, and *agent binding* (a, x) .

Basic definitions

$\text{Trk}(\mathcal{G}) \subseteq \text{St}^+$ denotes the set of all *tracks*, i.e., finite paths, of the CGS \mathcal{G} .

A *strategy* for \mathcal{G} is a partial function $f : \text{Trk}(\mathcal{G}) \rightarrow \text{Ac}$ mapping each track in its domain to an action.

When all the agents have associated strategies, they individuate a unique path in the underlying CGS \mathcal{G} , called the *play* of the game.

Informal meaning of $\langle\langle x \rangle\rangle\phi$, $[[x]]\phi$, and $(a, x)\phi$

Strategy quantification

- $\langle\langle x \rangle\rangle\phi$: “there exists a strategy x for which ϕ is true”;
- $[[x]]\phi$: “for all strategies x , it holds that ϕ is true”.

Agent binding

- $(a, x)\phi$: “ ϕ holds, when the agent a uses the strategy x ”.

Example: Failure is not an option

No failure property

“In a system \mathcal{S} built on three processes, α , β , and γ , the first two have to cooperate in order to ensure that the system itself never enters a failure state”.

Three different possible formalization in SL.

- $\langle\langle x \rangle\rangle \langle\langle y \rangle\rangle [[z]] (\alpha, x)(\beta, y)(\gamma, z)(G \neg fail)$: α and β have two strategies, x and y , respectively, that, independently of what γ decides, ensure that a failure state is never reached.
- $\langle\langle x \rangle\rangle [[z]] \langle\langle y \rangle\rangle (\alpha, x)(\beta, y)(\gamma, z)(G \neg fail)$: β can choose his strategy y in dependence of that one chosen by γ .
- $\langle\langle x \rangle\rangle [[z]] (\alpha, x)(\beta, x)(\gamma, z)(G \neg fail)$: α and β have a common strategy x to ensure the required property.

Multi-player Nash equilibrium

Nash equilibrium

Let \mathcal{G} be a CGS with the n agents $\alpha_1, \dots, \alpha_n$, each one having an its own LTL goal ψ_1, \dots, ψ_n .

We want to know if \mathcal{G} admits a Nash equilibrium, i.e., if there is for each agent α_i a “best” strategy x_i w.r.t. the goal ψ_i , once all other strategies are fixed.

$$\varphi_{NE} \triangleq \langle\langle x_1 \rangle\rangle \cdots \langle\langle x_n \rangle\rangle (\alpha_1, x_1) \cdots (\alpha_n, x_n) (\bigwedge_{i=1}^n (\langle\langle y \rangle\rangle (\alpha_i, y) \psi_i) \rightarrow \psi_i)$$

Intuitively, if $\mathcal{G} \models \varphi_{NE}$ then x_1, \dots, x_n form a Nash equilibrium, since, when an agent α_i has a strategy y that allows the satisfaction of ψ_i , he can use x_i instead of y , assuming that the remaining agents $\alpha_1, \dots, \alpha_{i-1}, \alpha_{i+1}, \dots, \alpha_n$ use $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n$.

Outline

- 1 Reasoning about Strategies
 - Syntax and Informal Semantics
 - Examples of Specifications
- 2 Decision problems
 - Model Checking
 - Satisfiability
- 3 Conclusion

Model checking via tree automata

General procedure

- 1 Given a model \mathcal{G} , construct a tree automaton $\mathcal{A}_{\mathcal{G}}$ recognizing the tree-unwinding of \mathcal{G} itself.
- 2 Given a specification φ , construct a tree automaton \mathcal{A}_{φ} recognizing all the tree models of φ itself.
- 3 Construct the product-automaton of $\mathcal{A}_{\mathcal{G}}$ and \mathcal{A}_{φ} and check for its emptiness.

Challenge here: The automaton \mathcal{A}_{φ} has to deal with the strategy quantifications of the formula φ .

Decidability of model checking

Key Idea (I): Every strategy quantification can be reduced to an action quantification, for each node of the tree-unwinding of the model.

Key Idea (II): Every action quantification can be handled locally on each node of the tree, by using the transition function of a tree automaton.

Thus, we reduce the model-checking problem to the emptiness of an exponential (in the size of the formula) parity tree automaton.
Hence, we obtain a **2EXPTIME** model-checking procedure for SL.

Recurrent domino problem

Definition

An *recurrent domino system* is a tuple $\mathcal{D} = \langle \mathbf{D}, H, V, t^* \rangle$ consisting of a finite non-empty set \mathbf{D} of *domino types*, two *horizontal* and *vertical matching relations* $H, V \subseteq \mathbf{D} \times \mathbf{D}$, and a *distinguished* tile type $t^* \in \mathbf{D}$.

The recurrent domino problem asks for an *admissible tiling* of $\mathbb{N} \times \mathbb{N}$, i.e., a function $\partial : \mathbb{N} \times \mathbb{N} \rightarrow \mathbf{D}$ that labels the plane consistently with H and V in such a way that the tile type t^* is repeated infinitely often on the first row of the grid.

The solution of the problem is known to be *undecidable* [Harel, '84]. In particular, it is Σ_1^1 -HARD.

An ordering on strategies (I)

Strategy ordering

Let \mathcal{G} be a CGS with two agents, α and β , and an atomic proposition p . We want to force the existence in \mathcal{G} of a strict partial order over strategies for α that has no upper bound.

$$x_1 < x_2 \triangleq \langle\langle y \rangle\rangle (\beta, y) ((\alpha, x_1)(Xp) \wedge (\alpha, x_2)(X\neg p)).$$

Note that $<$ is strict by definition.

An ordering on strategies (II)

Ordering sentence

$$\varphi^{ord} \triangleq \varphi^{unb} \wedge \varphi^{trn}.$$

$$1 \quad \varphi^{unb} \triangleq [[x_1]] \langle\langle x_2 \rangle\rangle x_1 < x_2;$$

$$2 \quad \varphi^{trn} \triangleq [[x_1]] [[x_2]] [[x_3]] (x_1 < x_2 \wedge x_2 < x_3) \rightarrow x_1 < x_3.$$

Intuitively, $<$ has no upper bound by Item 1 and is transitive by Item 2. Hence, it is a strict partial order without upper bound.

Note that \mathcal{G} requires an infinite number of actions in order to satisfy φ^{ord} . Moreover, we proved that \mathcal{G} cannot be turn-based.

Undecidability of satisfiability

Key Idea (I): By using the ordering sentence, we force the existence of two infinite chains of strategies for two players α and β .

Key Idea (II): The two chains represent two perpendicular sides of the grid $\mathbb{N} \times \mathbb{N}$, whose point are in bijection with the pairs of strategies for α and β .

We use the previous ideas to construct a reduction of the recurrent domino problem to the satisfiability of a particular SL formula.

Hence, we obtain that the satisfiability problem is Σ_1^1 -HARD.

Outline

- 1 Reasoning about Strategies
 - Syntax and Informal Semantics
 - Examples of Specifications

- 2 Decision problems
 - Model Checking
 - Satisfiability

- 3 Conclusion

Conclusion

In this work...

- we introduced SL, a new logic formalism for the temporal description of multi-player concurrent games, in which strategies are treated as first order objects;
- we show that the model-checking is 2EXPTIME-COMPLETE, improving the known non-elementary upper bound proved by K. Chatterjee and T.A. Henzinger and N. Piterman for their simpler logic;
- finally, we show that the satisfiability is highly undecidable, i.e., Σ_1^1 -HARD, so it is even not computably enumerable.

Thank you very much for your attention!
May the strategy be with you!