Reasoning about Strategies: From module checking to strategy logic

Aniello Murano

based on joint works with Fabio Mogavero, Giuseppe Perelli, Luigi Sauro, and Moshe Y. Vardi

Università degli Studi di Napoli "Federico II"

Luxembourg
September 23, 2013
Strategic Reasoning

Game Theory is a fruitful metaphor in the verification and synthesis of multi-agent systems, where agent behaviors are modeled by strategies in a game.

Plenty of modal logics for the specification of strategic reasonings have been introduced, but with a very limited power and no unifying framework.

Our aim

Looking for a powerful logic in which one can talk explicitly about the strategic behavior of agents in generic multi-player concurrent games.
An historical introduction to the framework

From monolithic to multi-agent systems

1. Closed systems verification: Model Checking
2. (System vs. Environment) open systems verification: Module Checking
3. Concurrent multi-agent system verification: ATL*
4. A multi-agent logic in which strategies are treated explicitly: Strategy Logic
Outline

1. From monolithic to multi-agent systems

2. Strategy Logic
   - Syntax and semantics
   - Interesting examples
   - Model-theoretic properties and expressiveness

3. Behavioral games
   - Why is S\textsubscript{L} so powerful?
   - Strategy dependence

4. Fragments of Strategy Logic
   - Semi-prenex fragments
   - Model-theoretic properties and expressiveness

4. At the end ...
Model checking

Historical development (1)

- Model checking: analyzes systems monolithically (system components plus environment) [Clarke & Emerson, Queille & Sifakis, ’81].

\[ M \models \varphi \]

Inputs

- The model \( M \) is a Kripke structure, i.e., a labeled-state transition graph.
- The specification \( \varphi \) is a temporal logic formula such as LTL, CTL or CTL*.
A closed system example: A drink dispenser-machine

\[ M : \]

- \( M = \langle AP, W, R, L, w_0 \rangle \)
- \( \varphi = \exists F \text{ Tea} \)
- \( M \text{ only makes internal non-deterministic choices} \)
- \( M \models \varphi \)

**Remark**

The system behavior can be represented by the unique three unwinding \( T_M \) of \( M \).
Module Checking

Historical development(2)
- Module checking: separates the environment from the system components, i.e., two-player game between system and environment [Kupferman & Vardi,’96-01].

\[ M \models_r \varphi \]

Inputs
- A module \( M \) is a Kripke structure with states partitioned in \( \text{Sys} \) and \( \text{Env} \) states.
- The specification \( \varphi \) is a temporal logic formula.

The problem
- Checking whether \( M \) is correct w.r.t. any possible behavior of the environment.
An open system example

- $M = \langle \text{AP}, \text{Sys}, \text{Env}, R, L, w_0 \rangle$
- $W = \text{Sys} \cup \text{Env}$
- $\text{Sys} \cap \text{Env} = \emptyset$

Remark: Everytime an Env state is met, the environment can disable some (but one) of its successors. Any possible behavior of the environment induces a different tree (i.e., a partial tree unwinding of $M$).

$T_M$ is a particular environment behavior.
An open system example

- $M = \langle AP, \text{Sys}, \text{Env}, R, L, w_0 \rangle$
- $W = \text{Sys} \cup \text{Env}$
- $\text{Sys} \cap \text{Env} = \emptyset$
- Always, at the Choose state, the environment makes a choice

Remark: Everytime an Env state is met, the environment can disable some (but one) of its successors. Any possible behavior of the environment induces a different tree (i.e., a partial tree unwinding of $M$).

$T_M$ is a particular environment behavior.
An open system example

- $M = \langle \mathcal{AP}, \text{Sys}, \text{Env}, R, L, w_0 \rangle$
- $W = \text{Sys} \cup \text{Env}$
- $\text{Sys} \cap \text{Env} = \emptyset$
- Always, at the Choose state, the environment makes a choice

$$\varphi = \exists F \text{Tea}$$
$$M \not\models_r \varphi$$
An open system example

- \( M = \langle \text{AP}, \text{Sys}, \text{Env}, R, L, w_0 \rangle \)
- \( W = \text{Sys} \cup \text{Env} \)
- \( \text{Sys} \cap \text{Env} = \emptyset \)
- Always, at the Choose state, the environment makes a choice
  - \( \varphi = \exists F \text{ Tea} \)
  - \( M \upharpoonright_r \varphi \)

**Remark**

- Everytime an Env state is met, the environment can disable some (but one) of its successors.
- Any possible behavior of the environment induces a different tree (i.e., a partial tree unwinding of \( M \)).
- \( T_M \) is a particular environment behavior.
Pro vs. Cons

Applications

- Module checking is very useful in open system verification. It allows to check whether a system is correct no matter how the environment behaves.

- It has been studied under perfect/imperfect information, hierarchical, infinite-state systems (pushdown, real-time), backwards modalities, graded modalities....
Pro vs. Cons

Applications

- Module checking is very useful in open system verification. It allows to check whether a system is correct no matter how the environment behaves.
- It has been studied under perfect/imperfect information, hierarchical, infinite-state systems (pushdown, real-time), backwards modalities, graded modalities....

Limitations

- Two-player game between system and environment.
- It is not powerful enough to be used in multi-player strategic reasoning.
Alternating-time Temporal Logic [Alur et al., ’02]

Historical development(3)

Alternating temporal reasoning: multi-agent systems (components individually considered), playing strategically [Alur et al.,’97-02].

**ATL**

Branching-time Temporal Logic with the strategic modalities \(<A>\) and \([A]\).

\(<A>\psi\): There is a strategy for the agents in \(A\) enforcing the property \(\psi\), independently of what the agents not in \(A\) can do.

**Example**

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

Aniello Murano
Università degli Studi di Napoli "Federico II"
Reasoning about Strategies: From module checking to strategy logic
A **concurrent game structure** is a tuple \( G = \langle \text{AP}, \text{Ag}, \text{Ac}, \text{St}, \lambda, \tau, s_0 \rangle \).

Intuitively

\( G \) is a Graph whose States \( \text{St} \) are labeled with Atomic Propositions \( \text{AP} \) and Transitions \( \tau \) are Agents’ Decision, i.e., Actions \( \text{Ac} \) taken by Agents \( \text{Ag} \).

**Strategy and Play**

A **strategy** is a function that maps each *history* of the game to an *action*. A **play** is a path of the game determined by the history of strategies.
The paper, rock, and scissor game

\[ \text{Ag} = \{ \alpha : Alice, \beta : Bob \} \]
\[ \text{St} = \{ s_i, s_\alpha, s_\beta \} \]
\[ s_i \text{ initial state} \]
\[ \text{AP} = \{ \text{win}_\alpha, \text{win}_\beta \} \]
\[ \text{Ac} = \{ P : \text{Paper}, R : \text{Rock}, S : \text{Scissor} \} \]

\[ D_i = \{(P, P), (R, R), (S, S)\} \]
\[ D_\alpha = \{(P, R), (R, S), (S, P)\} \]
\[ D_\beta = \{(R, P), (S, R), (P, S)\} \]
### Pro vs. Cons

<table>
<thead>
<tr>
<th>Pro</th>
<th>ATL* allows multi-agent strategic reasoning.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategies are treated only implicitly.</td>
</tr>
<tr>
<td>Quantifier alternation fixed to 1: either $\langle\langle \rangle\rangle[[]]$ or $[[]]\langle\rangle$.</td>
</tr>
</tbody>
</table>
Our contribution

Strategy Logic

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

Some useful fragments

We also consider a chain of syntactic fragments of SL that are *strictly more expressive* than ATL*, but *more tractable* than SL.
As for $\text{ATL}^*$, the underlying model is a CGs

Recall what is a CGs

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

...and its intuitive explanation

$G$ is a Graph whose States $\text{St}$ are labeled with Atomic Propositions $\text{AP}$ and Transitions $\tau$ are Agents’ Decision, i.e., Actions $\text{Ac}$ taken by Agents $\text{Ag}$. 
Syntax and semantics of SL

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

Syntax of SL

SL formulas are built as follows way, where $x$ is a variable and $a$ an agent.

$$\varphi ::= \text{LTL} \mid \langle\langle x \rangle\rangle\varphi \mid [[x]]\varphi \mid (a, x)\varphi.$$  

Semantics of SL

- $\langle\langle x \rangle\rangle\varphi$: “there exists a strategy $x$ for which $\varphi$ is true”.
- $[[x]]\varphi$: “for all strategies $x$, it holds that $\varphi$ is true”.
- $(a, x)\varphi$: “$\varphi$ holds, when the agent $a$ uses the strategy $x$”.
- LTL operators are classically interpreted on the resulting play.
Failure is not an option

Example (No failure property)

“In a system $S$ built on three processes, $\alpha$, $\beta$, and $\gamma$, the first two have to cooperate in order to ensure that $S$ never enters a failure state”.

Three different formalization in $S_L$.

1. $\langle\langle x \rangle\rangle\langle\langle y \rangle\rangle[[z]](\alpha, x)(\beta, y)(\gamma, z)(G \neg \text{fail})$: $\alpha$ and $\beta$ have two strategies, $x$ and $y$, ensuring that a failure state is never reached, independently of what $\gamma$ decides.

2. $\langle\langle x \rangle\rangle[[z]]\langle\langle y \rangle\rangle(\alpha, x)(\beta, y)(\gamma, z)(G \neg \text{fail})$: $\beta$ can choose his strategy $y$ dependently of that one chosen by $\gamma$.

3. $\langle\langle x \rangle\rangle[[z]](\alpha, x)(\beta, x)(\gamma, z)(G \neg \text{fail})$: $\alpha$ and $\beta$ have a common strategy $x$ to ensure the required property.
Multi-player Nash equilibrium

Example (Nash equilibrium)

Let $G$ be a game with the $n$ agents $\alpha_1, \ldots, \alpha_n$, each one having its own LTL goal $\psi_1, \ldots, \psi_n$. We want to know if $G$ admits a Nash equilibrium, i.e., if there is a “best” strategy $x_i$ w.r.t. the goal $\psi_i$, for each agent $\alpha_i$, once all other strategies are fixed.

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

Intuitively, if $G \models \phi_{NE}$ then $x_1, \ldots, x_n$ form a Nash equilibrium, since, when an agent $\alpha_i$ has a strategy $y$ that allows the satisfaction of $\psi_i$, he can use $x_i$ instead of $y$, assuming that the remaining agents $\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n$ use $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$. 
ATL* model-theoretic properties

Positive model-theoretic properties

- Invariance under bisimulation.
- Invariance under decision-unwinding.
- Bounded decision-tree model property.
## SL model-theoretic properties

### Negative model-theoretic properties
- **Non-invariance** under bisimulation.
- **Non-invariance** under decision-unwinding.
- **Unbounded** model property.

### Positive model-theoretic properties
- **Invariance** under state-unwinding.
- **State-tree** model property.
Expressiveness

Theorem

$SL$ is strictly more expressive than $ATL^*$. 

Explanation

- Unbounded quantifier alternation.
- Agents can be forced to share the same strategy.
A comparison

**Expressiveness**

SL is *more expressive* than ATL*.

**Computational complexities**

<table>
<thead>
<tr>
<th></th>
<th>ATL*</th>
<th>SL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model checking</td>
<td>$2\text{ExpTime-complete}$</td>
<td>“NONElementary-complete”</td>
</tr>
<tr>
<td>Satisfiability</td>
<td>$2\text{ExpTime-complete}$</td>
<td>Undecidable</td>
</tr>
</tbody>
</table>
A natural question

The question
Why is $S_L$ hard?

The answer
The choice of an action made by an agent in a strategy, for a given history of the game, may depend on the entire strategy of another agent, i.e., on its actions over all possible histories of the game.

An observation
Strategies are not synthesizable, since an agent, to have a chance to win, may need to forecast a possibly infinite amount of information about the behavior of an opponent.
Conterfactual dependence

\[ \varphi = [x]y \psi_1 \land \psi_2 \]

\[ \psi_1 = (\alpha, x)X p \leftrightarrow (\alpha, y)X \neg p \]

\[ \psi_2 = (\alpha, x)XX p \leftrightarrow (\alpha, y)XX p \]
Conterfactual dependence

\[ \phi = [x] \langle \langle y \rangle \rangle \psi_1 \land \psi_2 \]

\[ \psi_1 = (\alpha, x) X p \leftrightarrow (\alpha, y) X \neg p \]

\[ \psi_2 = (\alpha, x) X X p \leftrightarrow (\alpha, y) X X p \]
Conterfactual dependence

\[ \varphi = \llbracket x \rrbracket \langle \langle y \rangle \rangle \psi_1 \land \psi_2 \]

- \( \psi_1 = (\alpha, x)Xp \leftrightarrow (\alpha, y)X \neg p \)
- \( \psi_2 = (\alpha, x)XXp \leftrightarrow (\alpha, y)XXp \)
Conterfactual dependence

\[ \varphi = [x] \langle \langle y \rangle \rangle \psi_1 \land \psi_2 \]

\[ \psi_1 = (\alpha, x) X p \leftrightarrow (\alpha, y) X \neg p \]

\[ \psi_2 = (\alpha, x) X X p \leftrightarrow (\alpha, y) X X \neg p \]
Counterfactual dependence

\[ \varphi = [[x]] \langle \langle y \rangle \rangle \psi_1 \land \psi_2 \]

\[ \psi_1 = (\alpha, x) X p \leftrightarrow (\alpha, y) X \neg p \]

\[ \psi_2 = (\alpha, x) X X p \leftrightarrow (\alpha, y) X X p \]
Conterfactual dependence

\[
\varphi = \left[ [x] \langle y \rangle \right] \psi_1 \land \psi_2
\]

\[
\psi_1 = (\alpha, x) X p \leftrightarrow (\alpha, y) X \neg p
\]

\[
\psi_2 = (\alpha, x) X X p \leftrightarrow (\alpha, y) X X p
\]
Conterfactual dependence

\[ \varphi = [[x][y]]\psi_1 \land \psi_2 \]

\[ \psi_1 = (\alpha, x)Xp \leftrightarrow (\alpha, y)X\neg p \]

\[ \psi_2 = (\alpha, x)XXp \leftrightarrow (\alpha, y)XXp \]
Conterfactual dependence

\[ \varphi = [[x]] \langle \langle y \rangle \rangle \psi_1 \land \psi_2 \]
\[ \psi_1 = (\alpha, x) X p \leftrightarrow (\alpha, y) X \neg p \]
\[ \psi_2 = (\alpha, x) X X p \leftrightarrow (\alpha, y) X X p \]
Conterfactual dependence

\[ \varphi = \left[ [x] \langle \langle y \rangle \rangle \right] \psi_1 \land \psi_2 \]

\[ \psi_1 = (\alpha, x)X p \leftrightarrow (\alpha, y)X \neg p \]

\[ \psi_2 = (\alpha, x)XX p \leftrightarrow (\alpha, y)XX \neg p \]
Elementariness in strategies

**Behavioral property**
The quantification of a strategy is *behavioral* if the actions in a given history depend only on the actions of all other strategies on the same history.

**Behavioral semantics**
A formula is *behaviorally satisfiable* if it only needs behavioral strategies to be satisfied.

**Fact**
\( \text{ATL}^* \) is *behaviorally satisfiable*. 
Another question

The question

Is there any other syntactic fragment of $SL$ (strictly subsuming $ATL^*$) having a behavioral semantics?

Our answer

Yes! We obtain several fragments by using a prenex normal form for $SL$ and by putting different constraints on the use of bindings.
A quantification prefix is a sequence \( \varphi \) of quantifications in which each variable occurs once: \( \varphi = \langle x \rangle \langle y \rangle \langle z \rangle \langle w \rangle \).

A binding prefix is a sequence \( b \) of bindings such that each agent occurs once: \( b = (\alpha, x)(\beta, y)(\gamma, y) \).
Quantification and binning prefixes

A *quantification prefix* is a sequence $\mathfrak{q}$ of quantifications in which each variable occurs once: $\mathfrak{q} = [[[x]][[y]]⟨⟨z⟩⟩][[w]]$.

A *binding prefix* is a sequence $♭$ of bindings such that each agent occurs once: $♭ = (\alpha, x)(\beta, y)(\gamma, y)$.

A *goal* is a binding prefix $♭$ followed by an LTL formula.
Quantification and bining prefixes

A *quantification prefix* is a sequence $\varphi$ of quantifications in which each variable occurs once: $\varphi = [[x]][[y]]⟨⟨z⟩⟩[[w]]$.

A *binding prefix* is a sequence $♭$ of bindings such that each agent occurs once: $♭ = (α,x)(β,y)(γ,y)$.

A *goal* is a binding prefix $♭$ followed by an LTL formula.

By using a *prenex normal* form of a combination of goals, we identify a *chain of fragments*, which we name $SL[BG]$, $SL[DG / CG]$, and $SL[1G]$. 
Boolean-Goal Strategy Logic ($SL^{[BG]}$)

**Definition**

$SL^{[BG]}$ formulas are built inductively in the following way, where $\varnothing$ is a quantification prefix and $♭$ a binding prefix:

$$
\varphi ::= \text{LTL} \mid \varnothing \psi,
\psi ::= \♭ \varphi \mid \neg \psi \mid \psi \land \psi \mid \psi \lor \psi,
$$

where $\varnothing$ quantifies over all free variables of $\psi$.

- For $SL^{[CG]}$, we set $\psi ::= \♭ \varphi \mid \psi \land \psi$.
- For $SL^{[1G]}$, we set $\psi ::= \♭ \varphi$.

The expressiveness chain

$$
\text{ATL}^* < SL^{[1G]} < SL^{[CG]} < SL^{[BG]} \leq SL
$$
The behavioral results

The question

Which fragments of $SL$ have behavioral semantics?

Theorem

- $SL[BG]$ does not have behavioral semantics.
- $SL[CG]$ and $SL[1G]$ have behavioral semantics.
An overview

<table>
<thead>
<tr>
<th></th>
<th>Model checking</th>
<th>Satisfiability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SL</strong></td>
<td>“NonElementary-complete”</td>
<td>$\Sigma_1^1$-hard</td>
</tr>
<tr>
<td><strong>SL[BG]</strong></td>
<td>?</td>
<td>$\Sigma_1^1$-hard</td>
</tr>
<tr>
<td><strong>SL[CG]</strong></td>
<td>$2\text{ExpTime-complete}$</td>
<td>$\Sigma_1^1$-hard</td>
</tr>
<tr>
<td><strong>SL[1G]</strong></td>
<td>$2\text{ExpTime-complete}$</td>
<td>$2\text{ExpTime-complete}$</td>
</tr>
<tr>
<td><strong>ATL</strong>*</td>
<td>$2\text{ExpTime-complete}$</td>
<td>$2\text{ExpTime-complete}$</td>
</tr>
</tbody>
</table>
Model-theoretic properties

**SL[BG]** negative model-theoretic property

Unbounded model property.

**SL[CG]** negative model-theoretic properties

- Non-invariance under bisimulation.
- Non-invariance under decision-unwinding.

**SL[1G]** positive model-theoretic properties

- Invariance under bisimulation.
- Invariance under decision-unwinding.
- Bounded decision-tree model property.
In this talk

- We have introduced $\text{SL}$ as a logic for the temporal description of multi-player concurrent games, in which strategies are treated as first order objects.
- $\text{SL}$ model checking has a $\text{NONELEMENTARYTIME-COMPLETE}$ formula complexity and a $\text{PTIME-COMPLETE}$ data complexity.
- $\text{SL}$ satisfiability is highly undecidable, i.e., $\Sigma_1^1$-HARD.

- We have also introduced some fragments of $\text{SL}$, named $\text{SL[BG]}$, $\text{SL[CG]}$, and $\text{SL[1G]}$, all strictly more expressive than $\text{ATL}^*$.
- We have studied their model-theoretic properties. In particular, model-checking and satisfiability for $\text{SL[1G]}$ are no more complex than those for $\text{ATL}^*$, i.e., they are both $2\text{EXPTime-COMPLETE}$.
References


ATL* References


Module Checking References