Variations on the semantics of Strategy Logic

Nicolas Markey
CNRS – IRISA (Univ. Rennes, France)

(based on joint works with Patricia Bouyer, Patrick Gardy, and François Laroussinie)

68nqrt – February 1, 2018
Model checking and synthesis

system:

[http://www.embedded.com]

property:

$A \Box (\neg B.\text{overfull} \land \neg B.\text{dried\_up})$

model-checking algorithm

yes/no
Model checking and synthesis

system:

property:

[http://www.embedded.com]
Outline of the presentation

1. Introduction
   - Computation-Tree Logic (CTL) and Quantified CTL
   - Alternating-time Temporal Logic (ATL)

2. Strategy Logic
   - Model checking Strategy Logic (SL)
   - Semantics of SL
     - When does history start?
     - What do strategies know about each other?

3. Conclusions and future works
Outline of the presentation

1. Introduction
   - Computation-Tree Logic (CTL) and Quantified CTL
   - Alternating-time Temporal Logic (ATL)

2. Strategy Logic
   - Model checking Strategy Logic (SL)
   - Semantics of SL
     - When does history start?
     - What do strategies know about each other?

3. Conclusions and future works
Computation-Tree Logic (CTL) [CE81]

- atomic propositions: \( \bigcirc, \bigcirc, \ldots \)

[CE81] Clarke, Emerson. Design and Synthesis of Synchronization Skeletons... LOP, 1981.
Computation-Tree Logic (CTL) [CE81]

- atomic propositions: ○, ○, ...
- boolean combinators: ¬φ, φ ∨ ψ, φ ∧ ψ, ...

[CE81] Clarke, Emerson. Design and Synthesis of Synchronization Skeletons... LOP, 1981.
Computation-Tree Logic (CTL) [CE81]

- atomic propositions: $\circ$, $\bullet$, ...
- boolean combinators: $\neg \varphi$, $\varphi \lor \psi$, $\varphi \land \psi$, ...
- temporal modalities:
  - $X \varphi$ ("next $\varphi$")
  - $\varphi \mathcal{U} \psi$ ("$\varphi$ until $\psi$")

[CE81] Clarke, Emerson. Design and Synthesis of Synchronization Skeletons... LOP, 1981.
Computation-Tree Logic (CTL) [CE81]

- atomic propositions: \( \bigcirc, \bigcirc, \ldots \)
- boolean combinators: \( \neg \varphi, \varphi \lor \psi, \varphi \land \psi, \ldots \)
- temporal modalities:
  - \( X \varphi \) \( \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \) “next \( \varphi \)”
  - \( \varphi \ U \psi \) \( \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \) “\( \varphi \) until \( \psi \)”
  - true \( U \varphi \equiv F \varphi \) \( \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \) “eventually \( \varphi \)”
  - \( \neg F \neg \varphi \equiv G \varphi \) \( \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \) “always \( \varphi \)”

[CE81] Clarke, Emerson. Design and Synthesis of Synchronization Skeletons... LOP, 1981.
Computation-Tree Logic (CTL) [CE81]

- **atomic propositions:** \( \bigcirc, \bigcirc, \ldots \)
- **boolean combinators:** \( \neg \varphi, \varphi \lor \psi, \varphi \land \psi, \ldots \)
- **temporal modalities:**
  - \( X \varphi \) \( \quad \) \( \varphi \rightarrow \varphi \rightarrow \varphi \rightarrow \varphi \rightarrow \varphi \rightarrow \quad \) \( \text{“next } \varphi \text{”} \)
  - \( \varphi \mathbf{U} \psi \) \( \quad \) \( \varphi \rightarrow \varphi \rightarrow \psi \rightarrow \varphi \rightarrow \varphi \rightarrow \quad \) \( \text{“} \varphi \text{ until } \psi \text{”} \)
  - \( \text{true } \mathbf{U} \varphi \equiv \mathbf{F} \varphi \) \( \quad \) \( \varphi \rightarrow \varphi \rightarrow \varphi \rightarrow \varphi \rightarrow \varphi \rightarrow \quad \) \( \text{“eventually } \varphi \text{”} \)
  - \( \neg \mathbf{F} \neg \varphi \equiv \mathbf{G} \varphi \) \( \quad \) \( \varphi \rightarrow \varphi \rightarrow \varphi \rightarrow \varphi \rightarrow \varphi \rightarrow \quad \) \( \text{“always } \varphi \text{”} \)

- **path quantifiers:**
  - \( E \varphi \)
  - \( \mathbf{G} \varphi \)
  - \( A \varphi \)

[CE81] Clarke, Emerson. Design and Synthesis of Synchronization Skeletons... LOP, 1981.
Computation-Tree Logic (CTL) [CE81]

- atomic propositions: ○, ○, ...
- boolean combiners: ¬ϕ, ϕ ∨ ψ, ϕ ∧ ψ.
- temporal modalities:
  - Xϕ
  - ϕUψ
  - ϕ
  - true
  - ¬Fϕ

Examples

- **safety**: AG safe
  - only safe states will be reached
- **fairness**: AG AF fair
  - fair states will be visited infinitely many times
- **request-response**: AG(request ⇒ EF grant)
  - any request can eventually be granted

[CE81] Clarke, Emerson. Design and Synthesis of Synchronization Skeletons... LOP, 1981.
CTL model checking

In CTL, each temporal modality is in the immediate scope of a path quantifier.
CTL model checking

In CTL, each temporal modality is in the immediate scope of a path quantifier.

$$EF \equiv \text{some state is reachable}$$
In CTL, each temporal modality is in the immediate scope of a path quantifier.

\[ EF \equiv \text{some state is reachable} \]
In CTL, each temporal modality is in the immediate scope of a path quantifier.

\[ \text{EG}(\neg \mathbb{P} \land \text{EF} \mathbb{P}) \equiv \text{there is a path along which } \mathbb{P} \text{ is always reachable, but never reached} \]
In CTL, each temporal modality is in the immediate scope of a path quantifier.

\[ \text{EG}( \neg \square p \land \text{EF} \diamond p) \equiv \text{there is a path along which} \diamond p \text{ is always reachable, but never reached} \]
In CTL, each temporal modality is in the immediate scope of a path quantifier. 

\[ EG( \neg p \land EF p) \equiv \text{there is a path along which } p \text{ is always reachable, but never reached} \]
CTL model checking

In CTL, each temporal modality is in the immediate scope of a path quantifier.

Theorem ([CE81,QS82])

**CTL model checking is PTIME-complete.**

[CE81] Clarke, Emerson. Design and Synthesis of Synchronization Skeletons... LOP, 1981.

CTL model checking

In CTL, each temporal modality is in the immediate scope of a path quantifier.

Theorem ([CE81,QS82])

CTL model checking is PTIME-complete.

Remark ([KVW94])

CTL model checking on product structures is PSPACE-complete.

[CE81] Clarke, Emerson. Design and Synthesis of Synchronization Skeletons... LOP, 1981.
Quantified CTL  [ES84,Kup95,Fre01,DLM12]

QCTL extends CTL with propositional quantifiers

\[ \exists p. \varphi \text{ means that there exists a labelling of the model with } p \text{ under which } \varphi \text{ holds.} \]

Quantified CTL [ES84,Kup95,Fre01,DLM12]

QCTL extends CTL with propositional quantifiers

$\exists p. \varphi$ means that there exists a labelling of the model with $p$ under which $\varphi$ holds.

$$EF\bigcirc\land\neg\left(\exists p. \left[EF(\bigcirc\land p)\land EF(\bigcirc\land\neg p)\right]\right)$$

Quantified CTL [ES84,Kup95,Fre01,DLM12]

QCTL extends CTL with propositional quantifiers

\[ \exists p. \varphi \] means that there exists a labelling of the model with \( p \) under which \( \varphi \) holds.

\[ \text{EF} \, \Box \land \neg \left( \exists p. \left[ \text{EF}(\Box \land p) \land \text{EF}(\Box \land \neg p) \right] \right) \equiv \text{uniq}(\Box) \]

References:

Semantics of QCTL

- structure semantics:

\[ \models_s \exists p. \varphi \iff \models \varphi \]
Semantics of QCTL

- structure semantics:

\[ \models_s \exists p. \varphi \iff \models \varphi \]

- tree semantics:

\[ \models_t \exists p. \varphi \iff \models \varphi \]
Expressiveness of QCTL

- QCTL can “count”:

\[ \mathbf{E} \mathbf{X}_1 \varphi \equiv \mathbf{E} \mathbf{X} \varphi \land \forall p. [\mathbf{E} \mathbf{X}(p \land \varphi) \Rightarrow \mathbf{A} \mathbf{X}(\varphi \Rightarrow p)] \]

\[ \mathbf{E} \mathbf{X}_2 \varphi \equiv \exists q. [\mathbf{E} \mathbf{X}_1(\varphi \land q) \land \mathbf{E} \mathbf{X}_1(\varphi \land \neg q)] \]
Expressiveness of QCTL

- QCTL can “count”:

\[ \text{EX}_1 \varphi \equiv \text{EX} \varphi \land \forall p. [\text{EX}(p \land \varphi) \Rightarrow \text{AX}(\varphi \Rightarrow p)] \]
\[ \text{EX}_2 \varphi \equiv \exists q. [\text{EX}_1(\varphi \land q) \land \text{EX}_1(\varphi \land \neg q)] \]

- QCTL can express (least or greatest) fixpoints:

\[ \mu T. \varphi(T) \equiv \exists t. [\text{AG}(t \iff \varphi(t)) \land \forall t'. (\text{AG}(t' \iff \varphi(t')) \Rightarrow \text{AG}(t \Rightarrow t'))] \]
Expressiveness of QCTL

- QCTL can “count”:

  \[
  EX_1 \varphi \equiv EX \varphi \land \forall p. [EX(p \land \varphi) \Rightarrow AX(\varphi \Rightarrow p)]
  \]

  \[
  EX_2 \varphi \equiv \exists q. [EX_1(\varphi \land q) \land EX_1(\varphi \land \neg q)]
  \]

- QCTL can express (least or greatest) fixpoints:

  \[
  \mu T. \varphi(T) \equiv \exists t. [AG(t \iff \varphi(t)) \land \forall t'. (AG(t' \iff \varphi(t')) \Rightarrow AG(t \Rightarrow t'))]
  \]

Theorem

QCTL and MSO are equally expressive.
Decision problems for QCTL [LM14]

Theorem

Under the structure semantics:

- model checking is PSPACE-complete;
- satisfiability is undecidable.

Decision problems for QCTL [LM14]

Theorem

Under the structure semantics:
- model checking is PSPACE-complete;
- satisfiability is undecidable.

Under the tree semantics: for formulas with $k$ nested quantifiers,
- model checking is $k$-EXPTIME-complete;
- satisfiability is $(k+1)$-EXPTIME-complete.

Outline of the presentation

1. Introduction
   - Computation-Tree Logic (CTL) and Quantified CTL
   - Alternating-time Temporal Logic (ATL)

2. Strategy Logic
   - Model checking Strategy Logic (SL)
   - Semantics of SL
     - When does history start?
     - What do strategies know about each other?

3. Conclusions and future works
Games on graphs

Concurrent games

A concurrent game is made of
- a transition system;

![Diagram of a concurrent game with states q₀, q₁, and q₂]
Games on graphs

**Concurrent games**

A *concurrent game* is made of
- a transition system;
- a set of *agents* (or *players*);
Games on graphs

Concurrent games

A concurrent game is made of
- a transition system;
- a set of agents (or players);
- a table indicating the transition to be taken given the actions of the players.
Games on graphs

**Concurrent games**

A concurrent game is made of
- a transition system;
- a set of agents (or players);
- a table indicating the transition to be taken given the actions of the players.

**Turn-based games**

A turn-based game is a game where only one agent plays at a time.
Reasoning about open systems

**Strategies**

A *(pure) strategy* for a given player is a function telling which action to play depending on what has happened previously.
A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.
Reasoning about open systems

**Strategies**

A **(pure) strategy** for a given player is a function telling which action to play depending on what has happened previously.

**Example**

**Strategy for player**

![Diagram of a strategy for a player with nodes and arrows indicating decision points and interactions.](image-url)
Reasoning about open systems

**Strategies**

A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

**Example**

Strategy for player

alternately go to blue and green (starting with blue).
Reasoning about open systems

Strategies

A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

Example

Strategy for player

alternately go to and (starting with ).
Reasoning about open systems

**Strategies**

A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

**Example**

Strategy for player

alternately go to \(\bigcirc\) and \(\bigcirc\) (starting with \(\bigcirc\)).
Reasoning about open systems

**Strategies**

A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

**Example**

Strategy for player

alternately go to and (starting with ).
Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

\[ \langle A \rangle \varphi \] expresses that A has a strategy to enforce \( \varphi \).

Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

\( \langle A \rangle \varphi \) expresses that A has a strategy to enforce \( \varphi \).

Semantics of \( \langle A \rangle \varphi \)

Existential quantification (over strategies) implicitly includes a universal quantification (over outcomes):

\[
G, \Diamond \models \langle A \rangle \varphi \iff \exists \sigma_A. \forall \pi \in \text{Out}(\Diamond, \sigma_A). \pi \models \varphi.
\]

Temporal logics for games: ATL [AHK02]

ATL extends CTL with **strategy quantifiers**

\[ \langle A \rangle \varphi \] expresses that A has a strategy to enforce \( \varphi \).

Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

\[⟨⟨A⟩⟩ \varphi \] expresses that A has a strategy to enforce \( \varphi \).

Temporal logics for games: ATL [AHK02]

ATL extends CTL with **strategy quantifiers**

\[ \langle \langle A \rangle \rangle \varphi \] expresses that \( A \) has a strategy to enforce \( \varphi \).

Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

\( \langle A \rangle \varphi \) expresses that A has a strategy to enforce \( \varphi \).

Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

\(\langle A\rangle \varphi\) expresses that \(A\) has a strategy to enforce \(\varphi\).

Temporal logics for games: ATL [AHK02]

**ATL extends CTL with strategy quantifiers**

\( \langle \langle A \rangle \rangle \varphi \) expresses that \( A \) has a strategy to enforce \( \varphi \).

\[\begin{align*}
\langle \langle \bigcirc \rangle \rangle F & \equiv \langle \langle \Box \rangle \rangle F \\
\langle \langle \Box \rangle \rangle F & \equiv \langle \langle \bigcirc \rangle \rangle G(\langle \langle \Box \rangle \rangle F) \equiv \langle \langle \bigcirc \rangle \rangle G p
\end{align*}\]

Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

\[\langle A \rangle \varphi\] expresses that A has a strategy to enforce \(\varphi\).

\[\langle \Box \rangle F \quad \langle \Diamond \rangle F \quad \langle \Diamond \rangle G(\langle \Box \rangle F) \equiv \langle \Diamond \rangle G p\]
Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

\( \langle \langle A \rangle \rangle \varphi \) expresses that \( A \) has a strategy to enforce \( \varphi \).

Theorem ([AHK02])

*Model checking ATL is PTIME-complete.*

Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

$\langle A \rangle \varphi$ expresses that $A$ has a strategy to enforce $\varphi$.

Theorem ([AHK02])

Model checking ATL is PTIME-complete.

Remark ([LMO08])

In PTIME only if the transition table is given explicitly (size $|\text{Moves}| |\text{Agt}|$)

Outline of the presentation

1. Introduction
   - Computation-Tree Logic (CTL) and Quantified CTL
   - Alternating-time Temporal Logic (ATL)

2. Strategy Logic
   - Model checking Strategy Logic (SL)
   - Semantics of SL
     - When does history start?
     - What do strategies know about each other?

3. Conclusions and future works
Outline of the presentation

1. Introduction
   - Computation-Tree Logic (CTL) and Quantified CTL
   - Alternating-time Temporal Logic (ATL)

2. Strategy Logic
   - Model checking Strategy Logic (SL)
   - Semantics of SL
     - When does history start?
     - What do strategies know about each other?

3. Conclusions and future works
Strategy Logic [CHP07,MMV10]

Strategy logic

Explicit quantification over strategies + strategy assignment:

\[ \exists \sigma_A. \forall \sigma_B. \text{assign}(A \rightarrow \sigma_A, B \rightarrow \sigma_B). \varphi \]
Strategy Logic \[\text{[CHP07,MMV10]}\]

Strategy logic

Explicit quantification over strategies + strategy assignment:

$$\exists \sigma_A. \forall \sigma_B. \text{assign}(A \to \sigma_A, B \to \sigma_B). \varphi$$

- Quantification over strategies
- Assignment of strategies to players
- Path property

Example

- client-server interactions

\[ \exists \sigma_s. \exists \sigma_c. \text{assign}(S \rightarrow \sigma_S). (A G \text{ mutual exclusion } \land \]
\[ A G \bigwedge_{C} \text{assign}(C \rightarrow \sigma_c). A F \text{ access}_C) \]
Example

- client-server interactions
- existence of a pure boolean Nash equilibrium

\[ \exists \sigma_1 \ldots \exists \sigma_n. \text{assign}(A_1 \rightarrow \sigma_1, \ldots A_n \rightarrow \sigma_n). \]

\[ [\bigwedge_i ((\exists \sigma'_i . \text{assign}(A_i \rightarrow \sigma'_i). \varphi_i) \Rightarrow \varphi_i .)] \]
Strategy Logic

Example

- client-server interactions
- existence of a pure boolean Nash equilibrium
- existence of a dominant strategy for Player 1

$$\exists \sigma_1. \forall \sigma_2. \text{assign}(A_2 \rightarrow \sigma_2).$$

$$[(\exists \sigma'_1. \text{assign}(A_1 \rightarrow \sigma'_1). \varphi_1) \Rightarrow (\text{assign}(A_1 \rightarrow \sigma'_1). \varphi_1)]$$
Example

- client-server interactions
- existence of a pure boolean Nash equilibrium
- existence of a dominant strategy for Player 1
- existence of an admissible strategy for Player 1

\[
\exists \sigma_1. \neg \left( \exists \sigma'_1. \forall \sigma_2. \text{ assign}(A_2 \to \sigma_2). \right.
\]

\[
\left[ \text{ assign}(A_1 \to \sigma_1). \varphi_1 \Rightarrow \text{ assign}(A_1 \to \sigma'_1). \varphi_1 \right] \land
\]

\[
\exists \sigma_2. \text{ assign}(A_2 \to \sigma_2). \left( \neg \varphi_1 \land \text{ assign}(A_1 \to \sigma'_1). \varphi_1 \right)
\]
Model checking Strategy Logic

Theorem

*Strategy logic can be translated into QCTL.*
Strategy logic can be translated into QCTL.

- players has moves $m_1, ..., m_n$;
- from the transition table, we can compute the set $\text{Next}(\bullet, A, m_i)$ of states that can be reached from $\bullet$ when player $A$ plays $m_i$. 
Theorem

Strategy logic can be translated into QCTL.

SL can be translated as follows:

- encoding of $\exists \sigma. \psi$:
  $$\exists m_1^\sigma \exists m_2^\sigma \ldots \exists m_k^\sigma. \ A G (m_i^\sigma \iff \bigwedge \neg m_j^\sigma) \land \hat{\psi}$$
Model checking Strategy Logic

**Theorem**

*Strategy logic can be translated into QCTL.*

1. players has moves \( m_1, \ldots, m_n \);
2. from the transition table, we can compute the set Next(\( \square \), \( A \), \( m_i \)) of states that can be reached from \( \square \) when player \( A \) plays \( m_i \).

**SL can be translated as follows:**

- **encoding of** \( \exists \sigma . \psi \):  
  \[ \exists m_1^\sigma \exists m_2^\sigma \ldots \exists m_k^\sigma . \ A \ G (m_i^\sigma \iff \bigwedge \neg m_j^\sigma) \land \hat{\psi} \]

- **encoding of** assign\((\alpha)\). \( \varphi \) (for full binding \( \alpha : \text{Agt} \rightarrow \text{Strat} \)):  
  \[ A [G(q \land m_i^{\alpha(A)} \Rightarrow X \text{Next}(q, A, m_i^{\alpha(A)})) \Rightarrow \hat{\varphi}] \]
Theorem

Strategy logic can be translated into QCTL.

Theorem ([CHP07, MMV10, DLM12, LM13])

- Strategy-Logic model-checking is decidable.
- Strategy-Logic satisfiability is undecidable over concurrent games.
- Strategy-Logic satisfiability is decidable when restricted to turn-based games.

Outline of the presentation

1. Introduction
   - Computation-Tree Logic (CTL) and Quantified CTL
   - Alternating-time Temporal Logic (ATL)

2. Strategy Logic
   - Model checking Strategy Logic (SL)
   - Semantics of SL
     - When does history start?
     - What do strategies know about each other?

3. Conclusions and future works
What is the exact semantics? [BGM16]

$$\exists \sigma. \ AG(\text{assign}(A \rightarrow \sigma). \ \varphi)$$

What is the exact semantics? [BGM16]

\[ \exists \sigma. \textbf{AG}(\text{assign}(A \rightarrow \sigma). \varphi) \]

Strategy \(\sigma\) is selected here

What is the exact semantics? [BGM16]

\[ \exists \sigma. \textbf{AG} (\text{assign}(A \rightarrow \sigma). \varphi) \]
What is the exact semantics? [BGM16]

\[ \exists \sigma. \ AG(\text{assign}(A \rightarrow \sigma). \ \varphi) \]

Example

If \( \sigma \) prescribes to alternate between \( m_0 \) and \( m_1 \), starting with \( m_0 \):

What is the exact semantics? [BGM16]

$$\exists \sigma. \text{AG}(\text{assign}(A \rightarrow \sigma). \varphi)$$

**Example**

If $\sigma$ prescribes to alternate between $m_0$ and $m_1$, starting with $m_0$:

What is the exact semantics? [BGM16]

$$\exists \sigma. \ AG(\text{assign}(A \rightarrow \sigma). \ \varphi)$$

Example

If $\sigma$ prescribes to alternate between $m_0$ and $m_1$, starting with $m_0$:

History starts when strategy is selected.

What is the exact semantics? [BGM16]

\[ \exists \sigma. \text{AG}(\text{assign}(A \rightarrow \sigma). \varphi) \]

**Example**

If \( \sigma \) prescribes to alternate between \( m_0 \) and \( m_1 \), starting with \( m_0 \):

History starts when strategy is assigned.

This makes SL model checking undecidable.

What is the exact semantics? [BGM16]

$$\exists \sigma. \text{AG}(\text{assign}(A \rightarrow \sigma). \varphi)$$

Example

If $\sigma$ prescribes to alternate between $m_0$ and $m_1$, starting with $m_0$:

History starts when strategy is selected.

History starts when strategy is assigned.

What is the exact semantics? [BGM16]

\[ \exists \sigma. \ AG(\text{assign}(A \rightarrow \sigma). \ \varphi) \]

Example

If \( \sigma \) prescribes to alternate between \( m_0 \) and \( m_1 \), starting with \( m_0 \):

History starts when strategy is selected.

History starts when strategy is assigned.

What is the exact semantics? [BGM16]

\[ \exists \sigma. \text{AG}(\text{assign}(A \rightarrow \sigma). \varphi) \]

Example

If \( \sigma \) prescribes to alternate between \( m_0 \) and \( m_1 \), starting with \( m_0 \):

This is the semantics of Strategy Logic

History starts when strategy is selected.

History starts when strategy is assigned.

What is the exact semantics? [BGM16]

$$\exists \sigma. \ AG(\text{assign}(A \rightarrow \sigma). \varphi)$$

Example

If $\sigma$ prescribes to alternate between $m_0$ and $m_1$, starting with $m_0$:

History starts when strategy is selected.

This is the semantics of Strategy Logic

This makes SL model checking undecidable

History starts when strategy is assigned.

Encoding of a 2-counter machine

Strategy $\sigma$ encodes counter value $k$ if it plays $k$ times to $a$ before playing to $\bot$ (similarly for $\sigma'$).

Lemma Two strategies $\sigma$ and $\sigma'$ encode the same counter value iff, when entering the game above and applying these strategies, it holds $G(a \implies Xa) \land F[(a \land X\bot) \land \neg \Diamond \Diamond \bot] (\phi = \cdot)$.
Encoding of a 2-counter machine

Strategy $\sigma_\bigcirc$ encodes counter value $k$ if it plays $k$ times to $a$ before playing to $\bot$ (similarly for $\sigma_\square$).
Encoding of a 2-counter machine

Strategy $\sigma_\bigcirc$ encodes counter value $k$ if it plays $k$ times to $a$ before playing to $\perp$ (similarly for $\sigma_\bigbox$).

Lemma

Two strategies $\sigma_\bigcirc$ and $\sigma_\bigbox$ encode the same counter value iff, when entering the game above and applying these strategies, it holds

$$\text{G}(a \Rightarrow X\neg a) \land F[(a \land X\perp) \land \neg (\cdot\cdot) \neg XX\perp] \quad (\varphi=)$$
Encoding of a 2-counter machine

A strategy for $\circ$ encodes a run of the 2-counter machine;
A strategy for ◯ encodes a run of the 2-counter machine; At each step, Player □ can check correct values of counters:

\[
\begin{align*}
\Box s & \Rightarrow \exists \sigma. \text{EX} (\Box a \land \text{assign}(\Box \rightarrow \sigma).\varphi\_=) \land \\
\text{EX EX EX} s' & \land \text{EX} (\Box a \land \text{assign}(\Box \rightarrow \sigma).\varphi\_+) 
\end{align*}
\]
Outline of the presentation

1. Introduction
   - Computation-Tree Logic (CTL) and Quantified CTL
   - Alternating-time Temporal Logic (ATL)

2. Strategy Logic
   - Model checking Strategy Logic (SL)
   - Semantics of SL
     - When does history start?
       - What do strategies know about each other?

3. Conclusions and future works
Strategy dependences [MMPV11,GBM18]

Note
From this point on, we only consider formulas in SL[BG]:

\[
\text{assign} \ (A_j \mapsto \sigma_i). \ \phi
\]

ATL

[S1G]

SL[1G]

SL[DG]

SL[CG]

SL[BG]

SL


Strategy dependences [MMPV11, GBM18]

Note

From this point on, we only consider formulas in SL[BG]:

\[
\begin{align*}
\text{SL} & \quad \text{SL[BG]} \\
\text{SL[DG]} & \quad \text{SL[CG]} \\
\text{SL[1G]} & \quad \text{(Q}_i\sigma_i). \quad \text{assign}(A_j \rightarrow \sigma_i). \quad \varphi_{\text{LTL}}
\end{align*}
\]

Strategy dependences [MMPV11, GBM18]

Note

From this point on, we only consider formulas in SL[BG]:

\[ \text{SL} \quad \text{SL[BG]} \]
\[ \text{SL[DG]} \quad \text{SL(CG)} \quad \text{disjunction/conjunction of goals} \]
\[ \text{SL[1G]} \]
\[ \text{ATL} \]

\[ (Q_i \sigma_i). \quad \text{assign}(A_j \rightarrow \sigma_i). \varphi_{\text{LTL}} \]

Strategy dependences [MMPV11, GBM18]

Note

From this point on, we only consider formulas in SL[BG]:

SL

SL[BG]  boolean combination of goals

SL[DG]  disjunction/conjunction of goals

SL[CG]

SL[1G]

ATL

quantifier block  goal

Strategy dependences [MMPV11,GBM18]

\[ \forall \sigma_A \exists \sigma_B. \text{assign}(A \mapsto \sigma_A; B \mapsto \sigma_B). \varphi \]

Strategy dependences [MMPV11,GBM18]

∀σ_A∃σ_B. assign(A → σ_A; B → σ_B). ϕ

Strategy σ_B at node n may depend on the whole strategy σ_A.

Strategy dependences [MMPV11, GBM18]

∀σ_A ∃σ_B. assign(A → σ_A; B → σ_B). ϕ

Strategy σ_B at node n may depend on the whole strategy σ_A.

Strategy σ_B at node n only depends on σ_A along the branch to n.

Dependence maps (Skolemization)

Example

$v = \forall y. \exists z. \forall x_A. \forall x_B.$

\[
\bigvee \begin{cases}
\text{assign}(
\begin{array}{c}
\blacksquare \mapsto x_A; \\
\circ \mapsto y; \\
\blacklozenge \mapsto z
\end{array}
\). F \ p_1 \\
\text{assign}(
\begin{array}{c}
\blacksquare \mapsto x_B; \\
\circ \mapsto y; \\
\blacklozenge \mapsto z
\end{array}
\). F \ p_2
\end{cases}
\]
Dependence maps (Skolemization)

Example

\[ \varphi = \forall y. \exists z. \forall x_A. \forall x_B. \]
\[ \bigvee \left\{ \text{assign}(\square \mapsto x_A; \bigcirc \mapsto y; \Diamond \mapsto z). \quad \text{F } p_1 \right\} \]
\[ \text{assign}(\square \mapsto x_B; \bigcirc \mapsto y; \Diamond \mapsto z). \quad \text{F } p_2 \]

Dependence maps

Dependence maps select \( \exists \)-strategies from \( \forall \)-strategies:

\[ \theta : (\forall \rightarrow \text{Strats}) \rightarrow (\exists \rightarrow \text{Strats}) \]
\[ w \mapsto \theta(w) \]
Dependence maps (Skolemization)

Example

\[ \varphi = \forall y. \exists z. \forall x. \forall y. \]

\[ \bigvee \left\{ \text{assign}(\square \mapsto x_A; \bigcirc \mapsto y; \blacklozenge \mapsto z). \ F p_1 \right\} \]

\[ \bigvee \left\{ \text{assign}(\square \mapsto x_B; \bigcirc \mapsto y; \blacklozenge \mapsto z). \ F p_2 \right\} \]

Dependence maps

Dependence maps select \( \exists \)-strategies from \( \forall \)-strategies:

\[ \theta: (\forall \rightarrow \text{Strats}) \rightarrow (\exists \rightarrow \text{Strats}) \]

\[ w \mapsto \theta(w) \]

Then

\[ q_0 \models (Q_i \sigma_i). \Phi \iff \exists \theta. \forall w. q_0 \models_{\theta(w) \cup w} \Phi. \]
Dependence maps (Skolemization)

Example

\[ \varphi = \forall y. \exists z. \forall x_A. \forall x_B. \]

\[ \bigvee \left\{ \text{assign} (\square \mapsto x_A; \circ \mapsto y; \diamond \mapsto z). \ F \ p_1 \right\} \]

\[ \text{assign} (\square \mapsto x_B; \circ \mapsto y; \diamond \mapsto z). \ F \ p_2 \]

Dependence maps

Dependence maps select \( \exists \)-strategies from \( \forall \)-strategies:

\[ \theta : (\forall \to \text{Strats}) \to (\exists \to \text{Strats}) \]

\[ w \mapsto \theta(w) \]

Then

\[ q_0 \models (Q_i \sigma_i). \Phi \iff \exists \theta. \forall w. q_0 \models_{\theta(w) \cup w} \Phi. \]

\[ q_0 \not\models \varphi \text{ is (a priori) not equivalent to } q_0 \models \neg \varphi. \]
Dependence maps (Skolemization)

Example

\[ \varphi = \forall y. \exists z. \forall x_A. \forall x_B. \bigvee \left\{ \text{assign}(\square \mapsto x_A; \bigcirc \mapsto y; \Diamond \mapsto z) \right\} \]

\[ \text{F } p_1 \quad \text{F } p_2 \]

Dependence maps

- \( \forall \exists \) SL semantics:
  - no further constraints: existentially-quantified variables may depend on all universally-quantified variables.
Dependence maps (Skolemization)

Example

\[ \varphi = \forall y. \exists z. \forall x_A. \forall x_B. \]

\[ \bigvee \begin{cases} 
\text{assign}(\square \mapsto x_A; \bigcirc \mapsto y; \Diamond \mapsto z). \ F \ p_1 \\
\text{assign}(\square \mapsto x_B; \bigcirc \mapsto y; \Diamond \mapsto z). \ F \ p_2 
\end{cases} \]

Dependence maps

- classical SL semantics (\(\models^C\)):
  - depend on previously-quantified variables:
    
    if \(w(y) = w'(y)\), then \(\theta(w)(z) = \theta(w')(z)\)
Dependence maps (Skolemization)

Example

\[ \varphi = \forall y. \exists z. \forall x_A. \forall x_B. \]
\[ \bigvee \begin{cases} 
\text{assign(□} \mapsto x_A; \bigcirc \mapsto y; \blacklozenge \mapsto z). & F \ p_1 \\
\text{assign(□} \mapsto x_B; \bigcirc \mapsto y; \blacklozenge \mapsto z). & F \ p_2 
\end{cases} \]

Dependence maps

- elementary semantics \((\equiv^E)\):
  - depend on previously-quantified variables along current prefix:
    - for all \(n\), if \(w(y)(m) = w'(y)(m)\) for all \(m \preceq n\), then
    \[
    \theta(w)(z)(n) = \theta(w')(z)(n)
    \]
Dependence maps (Skolemization)

Example

\[
\varphi = \forall y. \exists z. \forall x_A. \forall x_B.
\]

\[
\bigvee \left\{ \begin{align*}
&\text{assign}([\square \mapsto x_A; \bigcirc \mapsto y; \Diamond \mapsto z]). & F p_1 \\
&\text{assign}([\square \mapsto x_B; \bigcirc \mapsto y; \Diamond \mapsto z]). & F p_2
\end{align*} \right\}
\]

Theorem

For any \( \varphi \in SL[BG] \):

\[
q_0 \models^C \varphi \iff q_0 \models \varphi \\
q_0 \models^E \varphi \Rightarrow q_0 \models^C \varphi
\]
Example

\[ \varphi = \forall y. \exists z. \forall x_A. \forall x_B. \]

\[ \bigvee \left\{ \begin{array}{l}
\text{assign}(\square \mapsto x_A; \bullet \mapsto y; \Diamond \mapsto z). \quad \text{F } p_1 \\
\text{assign}(\square \mapsto x_B; \bullet \mapsto y; \Diamond \mapsto z). \quad \text{F } p_2 \\
\end{array} \right\} \]

Theorem

For any \( \varphi \in \text{SL}[BG] \):

\[ q_0 \models^C \varphi \iff q_0 \models \varphi \quad q_0 \models^E \varphi \Rightarrow q_0 \models^C \varphi \]
Dependence maps (Skolemization)

**Example**

\[ \varphi = \forall y. \exists z. \forall x_A. \forall x_B. \]
\[ \bigvee \left\{ \text{assign}([\square \mapsto x_A; \bigcirc \mapsto y; \bigdiamond \mapsto z]). \ F p_1, \right. \]
\[ \left. \text{assign}([\square \mapsto x_B; \bigcirc \mapsto y; \bigdiamond \mapsto z]). \ F p_2 \right\} \]

- \( q_0 \models^C \varphi \): \( z \) plays to \( p_1 \) iff \( y \) plays to \( p_1 \).

**Theorem**

*For any \( \varphi \in SL[BG] \):

\[ q_0 \models^C \varphi \iff q_0 \models \varphi \quad q_0 \models^E \varphi \implies q_0 \models^C \varphi \]
Dependence maps (Skolemization)

**Example**

\[ \varphi = \forall y. \exists z. \forall x_A. \forall x_B. \]

\[ \bigvee \left\{ \begin{array}{l}
\text{assign(□} \leftrightarrow x_A; \bigcirc \leftrightarrow y; \Diamond \leftrightarrow z). \quad \mathbf{F} p_1 \\
\text{assign(□} \leftrightarrow x_B; \bigcirc \leftrightarrow y; \Diamond \leftrightarrow z). \quad \mathbf{F} p_2
\end{array} \right\} \]

- \( q_0 \models^C \varphi \): \( z \) plays to \( p_1 \) iff \( y \) plays to \( p_1 \).
- \( q_0 \not\models^E \varphi \): \( y \) only plays in \( q_1 \), so \( z \) will not depend on \( y \).

**Theorem**

For any \( \varphi \in \text{SL}[BG] \):

\[ q_0 \models^C \varphi \iff q_0 \models \varphi \quad q_0 \models^E \varphi \Rightarrow q_0 \models^C \varphi \]
Dependence maps (Skolemization)

Example

\[ \varphi = \forall y . \exists z . \forall x_A . \forall x_B . \]
\[ \bigvee \left\{ \begin{array}{l}
assign(\square \mapsto x_A ; \bigcirc \mapsto y ; \Diamond \mapsto z). \quad F \ p_1 \\
assign(\square \mapsto x_B ; \bigcirc \mapsto y ; \Diamond \mapsto z). \quad F \ p_2
\end{array} \right. \]

- \( q_0 \models^C \varphi \): \( z \) plays to \( p_1 \) iff \( y \) plays to \( p_1 \).
- \( q_0 \not\models^E \varphi \): \( y \) only plays in \( q_1 \), so \( z \) will not depend on \( y \).
- \( q_0 \models^E \neg \varphi \): because otherwise \( q_0 \models^C \neg \varphi \ldots \)

Theorem

For any \( \varphi \in SL[BG] \):

\[ q_0 \models^C \varphi \iff q_0 \models \varphi \]
\[ q_0 \models^E \varphi \Rightarrow q_0 \models^C \varphi \]
Elementary semantics

**Theorem**

For any $\varphi \in SL[1G]$, $q_0 \models^E \varphi$ iff $q_0 \not\models^E \neg \varphi$. 
Elementary semantics

Theorem

For any $\varphi \in SL[1G]$, $q_0 \models^E \varphi$ iff $q_0 \not\models^E \neg \varphi$.

Proof

From previous theorem:

$$q_0 \models^E \varphi \Rightarrow q_0 \models^C \varphi \Rightarrow q_0 \not\models^C \neg \varphi \Rightarrow q_0 \not\models^E \neg \varphi.$$
Elementary semantics

Theorem

For any $\varphi \in SL[1G]$, $q_0 \models^E \varphi$ iff $q_0 \not\models^E \neg \varphi$.

Proof

$\models^E \forall x. \exists y. \text{assign}(A \mapsto x, B \mapsto y). \; \psi$
Theorem

For any $\varphi \in SL[1G]$, $q_0 \models^E \varphi$ iff $q_0 \not\models^E \neg \varphi$.

Proof

proof_details
Theorem

For any $\varphi \in SL[1G]$, $q_0 \models^E \varphi$ iff $q_0 \not\models^E \neg \varphi$.

Proof

$\models^E \forall x. \exists y. \text{assign}(A \mapsto x, B \mapsto y). \psi$
Elementary semantics

Theorem

For any $\varphi \in SL[1G]$, $q_0 \equiv^E \varphi$ iff $q_0 \not\equiv^E \neg \varphi$.

Proof

$\equiv^E \forall x. \exists y. \text{assign}(A \mapsto x, B \mapsto y). \psi$

$\times A_\psi$

deterministic parity automaton
Elementary semantics

Theorem

For any $\varphi \in SL[1G]$, $q_0 \models^E \varphi$ iff $q_0 \not\models^E \neg \varphi$.

Proof

Player $P_\exists$ wins $\Rightarrow q_0 \models^E \varphi$; Player $P_\forall$ wins $\Rightarrow q_0 \not\models^E \neg \varphi$

Result follows by determinacy of turn-based parity games.
Elementary semantics

Theorem

For any $\varphi \in SL[1G]$, $q_0 \models^E \varphi$ iff $q_0 \not\models^E \neg \varphi$.

Proof

Model checking $SL[1G]$ with $\models^E$ is $2\text{EXPTIME}$-complete.
Beyond elementary dependences [GBM18]

Elementary semantics
Strategy $\sigma_B$ at node $n$ only depends on $\sigma_A$ along the branch to $n$. 

Beyond elementary dependences [GBM18]

Elementary semantics
Strategy $\sigma_B$ at node $n$ only depends on $\sigma_A$ along the branch to $n$.

Timeline semantics
Strategy $\sigma_B$ also depends on $\sigma_C$ along strict prefixes of $n$.

Beyond elementary dependences [GBM18]

∀σ
E
A
∃σ
E
B
∀σ
E
C

Timeline semantics
Strategy σ_B also depends on σ_C along strict prefixes of n.

Proposition
For any \( \varphi \in SL[BG] \),
\[ q_0 \models^E \varphi \Rightarrow q_0 \models^T \varphi \]

Beyond elementary dependences [GBM18]

Timeline semantics
Strategy $\sigma_B$ also depends on $\sigma_C$ along strict prefixes of $n$.

**Proposition**
For any $\varphi \in SL[BG]$, $\q_0 \models^E \varphi \Rightarrow \q_0 \models^T \varphi$

**Proposition**
For any $\varphi \in SL[1G]$, $\q_0 \models^E \varphi \iff \q_0 \models^T \varphi$
$\q_0 \models^E \neg \varphi \iff \q_0 \not\models^T \varphi$

Beyond elementary dependences

**Theorem**

*For a large fragment of SL[BG] (containing SL[CG] and SL[DG]),*

\[ q_0 \models^T \varphi \iff q_0 \not\models^T \neg \varphi. \]
Beyond elementary dependences

**Theorem**

For a large fragment of $SL[BG]$ (containing $SL[CG]$ and $SL[DG]$),

$q_0 \models_T \varphi \iff q_0 \not\models_T \neg \varphi$.

**Semi-stable sets**

For $s, f, g \in \{0, 1\}^n$, we let

$$\bar{s} : i \mapsto 1 - s(i)$$

$$f \succ g : \max\{f(i), g(i)\}$$

$$f \prec g : \min\{f(i), g(i)\}$$
Beyond elementary dependences

**Theorem**

For a large fragment of SL[BG] (containing SL[CG] and SL[DG]),
\[ q_0 \models^T \varphi \iff q_0 \not\models^T \neg \varphi. \]

**Semi-stable sets**

\( F \subseteq \{0, 1\}^n \) is semi-stable if
\[
\forall f, g \in F. \forall s \in \{0, 1\}^n. (f \land s) \lor (g \land \bar{s}) \in F \\
\text{or} \quad (f \land \bar{s}) \lor (g \land s) \in F
\]
Beyond elementary dependences

**Theorem**

For a large fragment of SL[BG] (containing SL[CG] and SL[DG]),

\[ q_0 \models^T \varphi \iff q_0 \not\models^T \neg \varphi. \]

**Semi-stable sets**

- any singleton is semi-stable;
- \{ (1, 1), (0, 0) \} is not semi-stable in \{0, 1\}^2.
- The complement of a semi-stable set is semi-stable.
Beyond elementary dependences

**Theorem**

For a large fragment of SL[BG] (containing SL[CG] and SL[DG]),

\[ q_0 \models^T \varphi \iff q_0 \not\models^T \neg \varphi. \]

**Proof (sketch of proof)**

- **SL[EG]:** fragment of SL[BG]:

  \[ (Q_i \sigma_i)_i . \text{Boolean-Combination}(G_1, \ldots, G_n) \]

  where **Boolean Combination** defines a semi-stable set.
Theorem

For a large fragment of SL[BG] (containing SL[CG] and SL[DG]),

\[ q_0 \models^T \varphi \iff q_0 \not\models^T \neg \varphi. \]

Proof (sketch of proof)

- **SL[EG]:** fragment of SL[BG]:

\[ (Q_i \sigma_i)_i, \text{ Boolean-Combination}(G_1, \ldots, G_n) \]

where Boolean Combination defines a semi-stable set.

Proposition

**SL[EG] contains SL[CG] and SL[DG], and is closed under negation.**
Beyond elementary dependences

Theorem

For a large fragment of SL[BG] (containing SL[CG] and SL[DG]),

\[ q_0 \models_T \varphi \iff q_0 \not\models_T \neg \varphi. \]

Proof (sketch of proof)

- extending the construction for SL[1G] with several goals:

  several goals ⇒ possibly several outcomes

\[
\forall x. \exists y. \begin{cases} 
\text{assign}(A \mapsto x; B \mapsto y). \varphi_1 \\
\text{assign}(A \mapsto y; B \mapsto x). \varphi_2 
\end{cases}
\]

... but this formula is not in SL[EG].
Beyond elementary dependences

**Theorem**

For a large fragment of SL[BG] (containing SL[CG] and SL[DG]),

\[ q_0 \models^T \varphi \iff q_0 \not\models^T \neg \varphi. \]

**Proof (sketch of proof)**

- extending the construction for SL[1G] with several goals:

several goals \(\Rightarrow\) possibly several outcomes

\[
\forall x. \exists y. \left[ \begin{array}{c}
\text{assign}(A \mapsto x; B \mapsto y). \varphi_1 \\
\leftrightarrow \\
\text{assign}(A \mapsto y; B \mapsto x). \varphi_2
\end{array} \right]
\]

... but this formula is not in SL[1G].
Beyond elementary dependences

Theorem

For a large fragment of $\text{SL[BG]}$ (containing $\text{SL[CG]}$ and $\text{SL[DG]}$),

$q_0 \models^T \varphi \iff q_0 \not\models^T \neg \varphi$.

Proof (sketch of proof)

- extending the construction for $\text{SL[1G]}$ with several goals:

  several goals $\Rightarrow$ possibly several outcomes

$$\forall x. \exists y. \left[ \begin{array}{l}
\text{assign}(A \mapsto x; B \mapsto y). \ \varphi_1 \\
\iff \\
\text{assign}(A \mapsto y; B \mapsto x). \ \varphi_2
\end{array} \right]$$
Beyond elementary dependences

**Theorem**

For a large fragment of SL[BG] (containing SL[CG] and SL[DG]),

\[ q_0 \models^T \varphi \iff q_0 \not\models^T \neg \varphi. \]

**Proof (sketch of proof)**

- extending the construction for SL[1G] with several goals:

  several goals \(\Rightarrow\) possibly several outcomes

\[ \forall x. \exists y. \left[ \begin{array}{l}
\text{assign}(A \mapsto x; B \mapsto y). \varphi_1 \\
\text{assign}(A \mapsto y; B \mapsto x). \varphi_2
\end{array} \right]\]

... but this formula is not in SL[EG].
Beyond elementary dependences

**Theorem**

For a large fragment of $SL[BG]$ (containing $SL[CG]$ and $SL[DG]$),

$$q_0 \models^T \varphi \iff q_0 \not\models^T \neg \varphi.$$

**Proof (sketch of proof)**

- solution for $SL[EG]$:
  - optimal behaviour along one outcome is independent of what happens along other outcomes;
  - inductively compute optimal sets of goals to be fulfilled depending on the situation.
Beyond elementary dependences

**Theorem**

For a large fragment of SL[BG] (containing SL[CG] and SL[DG]),

\[ q_0 \models^T \varphi \iff q_0 \not\models^T \neg \varphi. \]

**Proof (sketch of proof)**

- **solution for SL[EG]:**
  - optimal behaviour along one outcome is independent of what happens along other outcomes;
  - inductively compute optimal sets of goals to be fulfilled depending on the situation.

**Theorem**

Model checking SL[EG] is 2EXPTIME-complete (for \( \models^T \)).
Outline of the presentation

1. Introduction
   - Computation-Tree Logic (CTL) and Quantified CTL
   - Alternating-time Temporal Logic (ATL)

2. Strategy Logic
   - Model checking Strategy Logic (SL)
   - Semantics of SL
     - When does history start?
     - What do strategies know about each other?

3. Conclusions and future works
Conclusions and future works

Conclusions

- **Strategy Logic** is a very powerful logic for specifying properties of games on graphs;
- Understanding its exact semantics is difficult:
  - does it correspond to what users need/expect?

Future directions
- direct reduction to turn-based parity games for SL\[EG\];
- tree-automata approach for timeline semantics;
- model checking SL\[BG\] for \(|\equiv_T|\);
- and beyond SL\[BG\]?
- better understanding of semantics of SL;
- randomized strategies;
- extensions with quantities (e.g. time, energy, ...).
Conclusions and future works

**Conclusions**

- **Strategy Logic** is a very powerful logic for specifying properties of games on graphs;
- Understanding its exact semantics is **difficult**;
  \[ \sim \] does it correspond to what users need/expect?

**Future directions**

- direct reduction to turn-based parity games for SL[EG];
- tree-automata approach for timeline semantics;
- model checking SL[BG] for \( \models^T \); and beyond SL[BG]?
## Conclusions and future works

### Conclusions

- **Strategy Logic** is a very powerful logic for specifying properties of games on graphs;
- Understanding its exact semantics is **difficult**:
  - does it correspond to what users need/expect?

### Future directions

- **direct reduction** to turn-based parity games for SL[EG];
- **tree-automata approach** for timeline semantics;
- **model checking** $\mathsf{SL[BG]}$ for $\equiv^T$; and beyond $\mathsf{SL[BG]}$?
- Better understanding of semantics of SL;
- Randomized strategies;
- Extensions with quantities (e.g. time, energy, ...).