Composition and Synthesis via Game Structures

Giuseppe De Giacomo

Dipartimento di Informatica e Sistemistica
SAPIENZA Università di Roma
Rome, Italy

Introduction

Motivation

Example (Consider the following problems...)

- Conditional planning (even for temporally extended goals)
- Conditional planning in presence of (fully observable) exogenous events
- **Service/behavior/device composition**
- Agent planning programs, which mix planning and programming
- ...

There is a variety of behavior synthesis problems characterized by:

- Nondeterminism (of devilish nature!)
- Full observability

Key observation:

Sometimes we informally describe such problems as games between two players, where one player (the controller) tries to force that certain objectives no matter how other player (the environment) behave.

Introduction

Objectives:

- Take seriously the idea of modelling such synthesis problems as games among two contrasting agents.
- Develop a general framework for synthesis in AI based on two-player game structures.
- Develop reasoning/synthesis techniques leveraging on model-checking technologies.

In this talk:

- Introduce two-players game structures (2GSs)
- Introduce μ-calculus variant for expressing the ability of the controller to force the game to satisfy desired temporal properties.
- Device reasoning and synthesis techniques based on model checking of 2GSs.
- Apply such tools to a variety of problem and reconstruct solutions, in an optimal way wrt computational complexity.

Two-player Game Structures

Inspired by Pnueli's work on LTL synthesis by model checking (and aslo ATL).

- 2GS's are akin to transition systems used to describe the systems to be checked in Verification ...
- ... but with a substantial difference:

Two-player Game Structures

Inspired by Pnueli's work on LTL synthesis by model checking (and aslo ATL).

- 2GS's are akin to transition systems used to describe the systems to be checked in Verification ...
- ... but with a substantial difference:

  while a transition system describes the evolution of a system...

Two-player Game Structures

- A 2GS describes the joint evolution of two autonomous systems—the environment and the controller—running together and interacting at each step, as if engaged in a sort of game.

Two-player Game Structures

Formally, a two-player game structure (2GS) is a tuple:

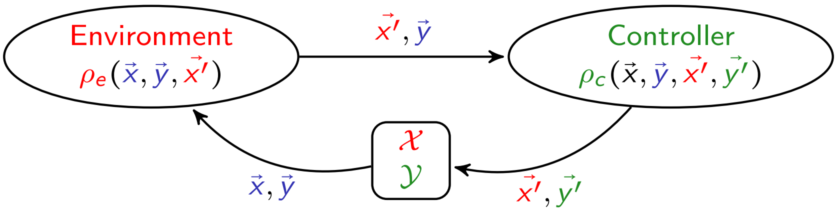
Definition (2GS)

$G = \langle \mathcal{X}, \mathcal{Y}, start, \rho_e, \rho_c \rangle$, where:

- $\mathcal{X} = \{x_1, \ldots, x_m\}$ is the set of environment (uncontrolled) variables ranging over finite domains;
- $\mathcal{Y} = \{y_1, \ldots, y_n\}$ are set of controller (controlled) variables ranging over finite domains;
- $start = \langle \vec{x}_o, \vec{y}_o \rangle$ is the initial state of the game.
- $\rho_e \subseteq \vec{X} \times \vec{Y} \times \vec{X}$ is the environment transition relation, which relates each game state to its possible successor environment states (or *moves*).
- $\rho_c \subseteq \vec{X} \times \vec{Y} \times \vec{X} \times \vec{Y}$ is the controller transition relation, which relates each game state and environment move to the possible controller replies.

2GS Transitions

2GS transitions:



- Uncontrolled ($\mathcal{X} = \{x_1, \ldots, x_n\}$) and controlled ($\mathcal{Y} = \{y_1, \ldots, y_m\}$) vars
- Environment assigns $\mathcal{X}$ vars (moves first),
- Controller sees results of environment'move and assigns $\mathcal{Y}$ vars
- Both have their own structural assumptions (constraints on execution)

Nondeterministic Planning Domains as a 2GS's

Example

Nondeterministic planning domain

$\mathcal{D} = \langle P, A, S_0, \rho \rangle$:

- $P = \{p_1, \ldots, p_n\}$ is a finite set of *domain propositions*;
- $A = \{a_1, \ldots, a_r\}$ is the finite set of *domain actions*;
- $S_0 \in 2^P$ is the *initial state*;
- $\rho \subseteq 2^P \times A \times 2^P$ is the *domain transition relation*.

Nondeterministic Planning Domains as a 2GS's

Example

Nondeterministic planning domain

$\mathcal{D} = \langle P, A, S_0, \rho \rangle$:

- $P = \{p_1, \ldots, p_n\}$ is a finite set of *domain propositions*;
- $A = \{a_1, \ldots, a_r\}$ is the finite set of *domain actions*;
- $S_0 \in 2^P$ is the *initial state*;
- $\rho \subseteq 2^P \times A \times 2^P$ is the *domain transition relation*.

Corresponding 2GS

$G_\mathcal{D} = \langle \mathcal{X}, \mathcal{Y}, start, \rho_e, \rho_c \rangle$:

- $\mathcal{X} = P$;
- $\mathcal{Y} = \{act\}$, with act ranging over $A \cup \{a_{init}\}$;
- $start = \langle S_0, a_{init} \rangle$;
- $\rho_e(S, a, S')$ iff $\rho(S, a, S') + \rho_e(S_0, a_{init}, S_0)$;
- $\rho_c(S, a, S', a')$ iff action a' is executable in S'
  (i.e., for some $\mathcal{S}'' \in 2^P$, $\rho(S', a', S'')$).

Goal Formulas

To express winning condition for the controller in 2GS's we introduce goal formulas.
For goal formulas, we use a variant of the μ-calculus interpreted over 2GS's.

Definition

Goal formulas

$$\Psi \leftarrow \varphi \mid Z \mid \Psi_1 \wedge \Psi_2 \mid \Psi_1 \vee \Psi_2 \mid \neg\Psi \mid \odot\Psi \mid \mu Z.\Psi \mid \nu Z.\Psi$$

Ingredients

- Atomic formulas φ of the form $(x_i = \bar{x}_i)$ and $(y_i = \bar{y}_i)$;
- Boolean operators;
- Special operator $\odot\Psi$ that expresses that the controller can force Ψ next.
- Least and greatest fixpoint constructs to capture sophisticated dynamic/temporal properties, defined by **induction** or **coinduction**.

Operator $\odot\Psi$

Definition ($\odot\Psi$ formal interpretation)

$$\langle\vec{x},\vec{y}\rangle \models \odot\Psi \text{ iff}$$
$$\exists\vec{x}'.\rho_e(\vec{x},\vec{y},\vec{x}') \land$$
$$\forall\vec{x}'.\rho_e(\vec{x},\vec{y},\vec{x}') \rightarrow \exists\vec{y}'.\rho_c(\vec{x},\vec{y},\vec{x}',\vec{y}') \text{ s.t. } \langle\vec{x}',\vec{y}'\rangle \models \Psi.$$

$\odot\Psi$ intuitive meaning

For every move $\vec{x}$ of the environment from the game state $\langle\vec{x},\vec{y}\rangle$, there is a move $\vec{y}'$ of controller such that in the resulting state of the game $\langle\vec{x}',\vec{y}'\rangle$ the property Ψ holds.

Note: in μ-calculus such alternation of quantification (universal for the environment) and (existential for the controller) can be easily expressed!

Examples of Goal Formulas

Example (liveness: eventually goal)

A standard conditional planning goal: **reach** a desired state of affairs can be expressed as

$$\diamond goal \doteq \mu Z.\ goal \lor \odot Z.$$

Examples of Goal Formulas

Example (liveness: eventually goal)

A standard conditional planning goal: **reach** a desired state of affairs can be expressed as

$$\Diamond goal \ \doteq\ \mu Z.\ goal \vee \odot Z.$$

Example (safety: always *goal*)

Now assume to have a domain with exogenous actions then **maintaining** a property *goal* still in spite of environment moves can be expressed:

$$\boxdot goal \ \doteq\ \nu Z. goal \wedge \odot Z.$$

Examples of Goal Formulas

Example (liveness: eventually goal)

A standard conditional planning goal: **reach** a desired state of affairs can be expressed as

$$\Diamond goal \ \doteq\ \mu Z.\ goal \vee \odot Z.$$

Example (safety: always *goal*)

Now assume to have a domain with exogenous actions then **maintaining** a property *goal* still in spite of environment moves can be expressed:

$$\boxdot goal \ \doteq\ \nu Z. goal \wedge \odot Z.$$

Example (fairness: infinitely often *goal*)

In the same setting, we may be content with a strategy to force the game so that it is **always** the case that **eventually** a state where *goal* holds is reached.

$$\boxdot\Diamond goal \ \doteq\ \nu Z_1.(\mu Z_2.((goal \wedge \odot Z_1) \vee \odot Z_2))$$

Service Composition

Example

Composition

Given a target service $\mathcal{S}_0$ and available service $\mathcal{S}_1, \ldots, \mathcal{S}_n$ with $\mathcal{S}_i = \langle A, S_i, s_{i0}, \delta_i, F_i \rangle$, check whether there exists a composition (and if so return it).

Service Composition

Example

Composition

Given a target service $\mathcal{S}_0$ and available service $\mathcal{S}_1, \ldots, \mathcal{S}_n$ with $\mathcal{S}_i = \langle A, S_i, s_{i0}, \delta_i, F_i \rangle$, check whether there exists a composition (and if so return it).

Simulation

Given a target service $\mathcal{S}_0$ and available service $\mathcal{S}_1, \ldots, \mathcal{S}_n$ with $\mathcal{S}_i = \langle A, S_i, s_{i0}, \delta_i, F_i \rangle$, check whether $\mathcal{S}_1, \ldots, \mathcal{S}_n$ can simulate (forever) $\mathcal{S}_0$ and (and if so return the "simulation strategy").

2GS for Service Composition

Example (2GS for Service Composition)

$G_{\mathcal{D}} = \langle \mathcal{X}, \mathcal{Y}, start, \rho_e, \rho_c \rangle$:

- $\mathcal{X} = \{st_0, st_1, \ldots, st_n, act, err\}$; with st_i ranging over S_i, act ranging over A and err over booleans;
- $\mathcal{Y} = \{srv\}$, with srv ranging over $1, \ldots, n$;
- $\rho_e(st_0, st_1, \ldots, st_n, act, err, srv, st'_0, st'_1, \ldots, st'_n, act', err')$ for
  - $err = false$, $srv = i$ and for at least one st'_i, we have $\delta_i(st_i, act, st'_i)$:
    - $\delta_i(st_i, act, st'_i)$;
    - $st'_j = st_j$ for $j = 1, \ldots, n$ and $j \neq i$;
    - $st'_0 = \delta_0(st_0, act)$ (recall that the target service is deterministic);
    - act' s.t. $\delta_0(st'_0, act')$ defined (NB: note this is the next step in the target!);
    - $err' = false$;
  - or $err = false$, $srv = i$ and for no st'_i $\delta_i(st_i, act, st'_i)$; or if already $err = true$:
    - $st'_j = st_j$ and $j = 1, \ldots, n$
    - $st'_0 = st_0$ (recall that the target service is deterministic);
    - $act' = act$;
    - $err' = true$
  - plus a suitable treatment of the starting state $start$.
- $\rho_c(st_0, \ldots, err, srv, s'_0, \ldots, err', srv')$ for $srv' = 1, \ldots, n$.

Goal Formulas for Service Composition

Example (Goal Formulas for Service Composition)

The goal formula requires the to always maintain the following condition ϕ true:

$$\phi \doteq \neg err \wedge (F_0 \rightarrow F_1 \wedge \ldots \wedge F_n)$$

That is:

$$\boxdot\phi \doteq \nu Z.\phi \wedge \odot Z.$$

This is a so called safety formula.

Reasoning (Model Checking) on 2GS

Theorem

Checking a goal formula Ψ over a game structure $G = \langle \mathcal{X}, \mathcal{Y}, start, \rho_e, \rho_c \rangle$ can be done in time

$$O((|G| \cdot |\Psi|)^k)$$

where $|G|$ denotes the number of game states of G plus $|\rho_e| + |\rho_c|$, $|\Psi|$ is the size of formula Ψ (considering propositional formulas as atomic), and k is the number of nested fixpoints sharing the same free variables in Ψ.

Observation

In fact we can easily adapt standard model checking algorithms for μ-calculus:

- *Note that while we use $\odot\Psi$ operator, which, though more sophisticated than in standard μ-calculus $\langle\Psi\rangle$, in order to evaluate it we only needs local checks.*

Examples (Cont.)

Example (liveness: eventually goal)

A standard conditional planning goal: $\diamond goal \doteq \mu Z.\, goal \vee \odot Z$.

Can be done in linear time in the size of the 2GS G, i.e., $2^{|G|}$ wrt a compact representation of G (Problem is known to be EXPTIME-complete.)

Examples (Cont.)

Example (liveness: eventually goal)

A standard conditional planning goal: $\Diamond goal \doteq \mu Z.\ goal \vee \odot Z$.

Can be done in linear time in the size of the 2GS G, i.e., $2^{|G|}$ wrt a compact representation of G (Problem is known to be EXPTIME-complete.)

Example (safety: always *goal*)

Maintaining a property *goal* in spite of environment moves:
$\boxdot goal \doteq \nu Z. goal \wedge \odot Z$.

Can be done in linear time in the size of the 2GS G, i.e., $2^{|G|}$ wrt a compact representation of G. (Problem also is known to be EXPTIME-complete.)

Examples (Cont.)

Example (liveness: eventually goal)

A standard conditional planning goal: $\Diamond goal \doteq \mu Z.\ goal \vee \odot Z$.

Can be done in linear time in the size of the 2GS G, i.e., $2^{|G|}$ wrt a compact representation of G (Problem is known to be EXPTIME-complete.)

Example (safety: always *goal*)

Maintaining a property *goal* in spite of environment moves:
$\boxdot goal \doteq \nu Z. goal \wedge \odot Z$.

Can be done in linear time in the size of the 2GS G, i.e., $2^{|G|}$ wrt a compact representation of G. (Problem also is known to be EXPTIME-complete.)

Example (fairness: infinitely often *goal*)

Force the game so that it is always the case that eventually a state where *goal* holds is reached: $\boxdot\Diamond goal \doteq \nu Z_1.(\mu Z_2.((goal \wedge \odot Z_1) \vee \odot Z_2))$

Can be done in linear time in the size of the 2GS G, i.e., $2^{|G|^2}$ wrt a compact representation of G. (Problem is EXPTIME-complete.)

Synthesis

Strategies

A controller strategy is a partial function

$$f : (\vec{X} \times \vec{Y})^{+} \times \vec{X} \mapsto \vec{Y}$$

such that for every sequence $\lambda = \langle \vec{x}_0, \vec{y}_0 \rangle \cdots \langle \vec{x}_n, \vec{y}_n \rangle$ and every $\vec{x}' \in \vec{X}$ such that $\rho_e(\vec{x}_n, \vec{y}_n, \vec{x}')$ holds, it is the case that $\rho_c(\vec{x}_n, \vec{y}_n, \vec{x}', f(\lambda, \vec{x}'))$ applies.

Extracting winning strategy from model checking witness

- Model checking algorithms provide a witness of the checked property.
- The witness consists of a labeling of the game structure produced during the model checking process.
- From labelled game states, one can read how the controller is meant to react to the environment at each step in order to fulfill the formulas that label the state itself, and from this, define a strategy to fulfill the goal formula.

Implementation

What's available off-the-shelf

- There are a few model checker for μ-calculus – but none very optimized.
- Most of them do (symbolically) search backward (typical in model checking), but interestingly some work forward ("local model checking").
- For formulas without nested fixpoints one can use ATL model checkers such as MCMAS. But notice that, e.g., fairness cannot be expressed!
- For some of the most prominent 2-nested fixpoints properties one can use Pnueli's TLV also based on synbolic methods (used for GR(1) LTL –strong fairness constraints).

In general, more work has to be done, but quite promising: we can leverage on available model checking techniques!

Conclusion

Summary

- 2GS is a powerful framework to express and solve sophisticates synthesis problems ...
- ... such as: conditional planning, planning against adversaries, synthesis for sophisticated temporal properties, composition/repurposing of available behaviors, ...
- Solvers can be readily implemented: either using directly off-the-shelf tools, or by developing tools using available model checking technology.

Conclusion

Summary

- 2GS is a powerful framework to express and solve sophisticates synthesis problems ...
- ... such as: conditional planning, planning against adversaries, synthesis for sophisticated temporal properties, composition/repurposing of available behaviors, ...
- Solvers can be readily implemented: either using directly off-the-shelf tools, or by developing tools using available model checking technology.

Personal note

I'd like to thank Amir Pnueli [Apr. 22, 1941 Nov. 2, 2009].

- I met him in June 2005 at a Dagstuhl seminar *[Synthesis and Planning organized by Kautz, Thomas, Vardi]*.
- We talked about service/behavior composition, and he suggested me to look into LTL synthesis via model checking.
- In June 2006 he visited Rome and gave a PhD course on LTL synthesis, including synthesis by model checking *[Fabio Patrizi's PhD Thesis, 2009]*.
- It was an extremely fruitful visit as this AAAI-10 paper (and several papers before this) testifies.