Solving infinite games and reactive synthesis

Zhou Jie, MSc AI, 31384579, jz5n19@soton.ac.uk

***Abstract*—In this review, we will first introduce the problem of synthesizing a reactive system. Since it used the model of infinite game on finite graph in the synthesis process, we then briefly go through the related basic concepts and the algorithms to solve these games in section II. In section III, we will describe the linear temporal logic (LTL) used in the specification on which the synthesis process is based and break down the synthesis process in several steps in detail. In the section IV, we will summarize the relationship between reactive synthesis and infinite game solving.**

I. Introduction

A reactive system has two characteristics: nonterminating and interactive. The former, like the operating system [1] needs to continuously process various input task. The latter emphasize the interaction between system and environment.
The synthesis process of such system can be seen as a game between two players: system and environment. The result of the synthesis is either implementation or unrealizable. Implementation means the system satisfies the specification, no matter what the environment does. While the system is unrealizable if the specification is falsified by the environment. [2] Since the synthesis process can be reduced to solving a game, we will first introduce the game in section II and explain the relationship with reactive synthesis in section III

II. Infinite games over finite graph

A. Related concepts

① Infinite games over finite graph

The game is played on an arena defined as a triple [1]

$$A = (V_0, V_1, E)$$

where V_0 and V_1 represent the set of positions of player 0 and 1 respectively, and they consist of all the positions V in the arena, i.e., $V = V_0 \cup V_1$. E represents the set of edges between positions, every position has at least one outgoing edge, i.e., a directed edge that the position is the origin.

The game is called infinite game because of the play of the game is an infinite sequence of positions

$$\pi = P_0 P_1 P_2 \ldots \in V^\omega$$

A play progresses in the following way: a token is placed on a position, the player who owns that position should move the token to the next position along the direction of the edge. Then the player who owns the successive position will continue moving the token in the same way, and so on.

A strategy f_σ for player σ is a function to decide what's the next position should the token go when the token is currently on the position of player σ. A play $\pi = P_0 P_1 P_2 \ldots$ is said to be conform to strategy f_σ if $P_{i+1} = f_\sigma(P_i)$.

② Winning condition

There are three types of infinite games: safety/reachability game, Buchi/co-Buchi game and parity game. In addition to the arena, different types of games have different definition components.

The safety/reachability game is defined as

$$G = (A, S)$$

where A is arena and S is called a safe set of positions. In the safety game, player 0 wins a play if the positions of the play is always in the safe set, otherwise player 1 wins.

The Buchi/Co-Buchi game is defined as

$$G = (A, F)$$

where A is arena and F is a proper subset of V. Player 0 wins a play if the set of positions that occur infinitely often in it has no intersection with F, otherwise player 1 wins.

The parity game is defined as

$$G = (A, \alpha)$$

where A is arena and α is a coloring function whose input is a position and the output is an integer representing priority of the position. Player 0 is said to be won a play if the largest priority occurs infinitely in the play is even [3], otherwise, player1 wins.

③ determinacy

Determinacy refers to a characteristic of the game, that is, every position in the arena belongs to the 'winning region' of either player 1 or player 2, i.e., no position missed. In order to know what is the winning region of player σ above, we need to define what is p-winning for a strategy f_σ first.

A strategy f_σ is called p-winning for player σ and position p must meet the following three condition, that is, if all plays that conforms to f_σ and that start in p are won by player σ [4]. So the winning region of player σ is a set of the positions with p-winning strategy f_σ there. If every position in the arena is covered by the winning region of a player, the game is determined.

Some strategies have a special property called memoryless. It means at position p the output of the strategy function f_σ , i.e., the successive position depends only on current position p and not affected by the preceding positions in the play sequence. Furthermore, if there is a memoryless strategy make a certain player win the game at every position of the arena, the game is called memoryless determined

B. *Solving games*

Given a starting position, solving a game means to decide which player has a strategy to win, i.e., returning the winning regions of the two players.

①Solving Safety game

For the safety game, the winning region for player 1 W_1 can be computed as below [4]

$$W_1 = Attr_1(V \backslash S, G)$$

where $V \backslash S$ is the complement of S, and $Attr_1(V \backslash S, G)$ is called the attractor set, it is the greatest proper subset of V such that player 1 can use a strategy to attract the token from any position of it to $V \backslash S$. [7] Since the safety game is determined, the winning region for player 0 W_0 is the complement of W_1.

The computation of the attractor set or attractor construction is by adding the states iterably until no more states can be added

②Solving Buchi game

For the Buchi game, the winning region for player 0 W_0 can be computed as below [4]

$$W_0 = Attr_0(Recur_0(G), G)$$

where $Recur_0(G)$ means the recurrent set of positions in the play. And the computation of recurrent set is by the process of recurrence construction. Since the Buchi game is also determined, once the winning region for player 0 W_0 is computed, the winning region for player 1 W_1 is also clear.

③Solving Parity game

Parity game can be solved by using McNaughton's Algorithm. [4] It's done in a way that recursively descent the highest color/priority in the remaining arena. The pseudo code below describes the algorithm [4]

Define *McNaughton*(*G*)

c := highest color in *G*
if *c* = 0 or *V* = ∅ **then**
  return (*V*, ∅)
set σ to *c* **mod** 2
set $W_{1-\sigma}$ to ∅
repeat
  $G' := G \setminus Attr_\sigma(\alpha^{-1}(c), G)$
  $(W'_0, W'_1) := McNaughton(G')$
  if $(W'_{1-\sigma} = \varnothing)$ **then**
    $W_\sigma := V \setminus W_{1-\sigma}$
    return (W_0, W_1)
  $W_{1-\sigma} := W_{1-\sigma} \cup Attr_{(1-\sigma)}(W'_{1-\sigma}, G)$
  $G := G \setminus Attr_{(1-\sigma)}(W'_{1-\sigma}, G)$

III. Reactive System Synthesis

A. *Overview*

One of the most important applications of parity game solving is controller synthesis of reactive system.

The synthesis is based on the specification given by a fragment of linear temporal logic (LTL) [2]. The synthesis process begins by constructing a game played by the players of system and environment. They start from an initial state and continue playing or moving to the next state according to some transition relations. If the specification is implemented, it means the specification is satisfied by the system and there will be an automaton extracted. Running the synthesized automaton means moving along a sequence of states and guiding the system to follow and take related actions. If the specification is unrealizable, it means the specification is falsified by the environment.

B. *LTL*

As stated above, the specification is expressed in a formal language called linear temporal logic (LTL), which is a fragment of first order logic. It can express the sense of future path that represents state transition. This explains the word 'temporal' in the name.

①Syntax and semantics of the full LTL

Syntactically, A formula of LTL is made of atomic propositions, logical connective: negation (¬), disjunction (∨), and temporal operators: "next" ○, "until" (*U*). In another word, if ψ and φ are LTL formulas then ¬ψ, φ ∨ ψ, ○ ψ, and φ *U* ψ are LTL formulas [6]. Furthermore, from these basic elements we can derive more propositions and operators:

Symbol	Meaning	Equivalent Components
True	True	$\varphi \vee \neg\varphi$
False	False	¬True
$\varphi \wedge \psi$	Conjunction	$\neg(\neg\varphi \vee \neg\psi)$
$\varphi \Rightarrow \psi$	Implication	$\neg\varphi \vee \psi$
$\varphi \Leftrightarrow \psi$	Equivalence	$(\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$
$\Diamond \varphi$	Eventually	True *U* φ
$\Box \varphi$	Always	$\neg\Diamond\neg\varphi$

The semantic of a LTL formula is about being satisfied by a sequence σ of true values of atomic propositions. We denote sequence σ satisfies formula φ, at position *i* as $\sigma, i \models \varphi$. [2] For example:

$$\sigma, i| = \varphi 1 \vee \varphi 2, \qquad \text{iff } \sigma, i| = \varphi 1 \text{ or } \sigma, i| = \varphi 2$$

②Generalized reactivity(1) formula

In the motion planning synthesis, a special form of LTL formula is used as the specification:

$$\varphi = (\varphi_e \Rightarrow \varphi_s)$$

where φ_e and φ_s represent the behaviors of environment and the system respectively. They both have the following structure [2]:

$$\varphi_e = (\varphi_i^e \wedge \varphi_t^e \wedge \varphi_g^e);\ \varphi_s = (\varphi_i^s \wedge \varphi_t^s \wedge \varphi_g^s)$$

The subscripts of *i*, *t*, and *g* indicate that they are evolved in a temporal order manner, where *i* represents the initial values of propositions (of environments and system), *t* represents the

transition of states (of environments and system), and g represents the goal assumption of environment and the goal specification of system.

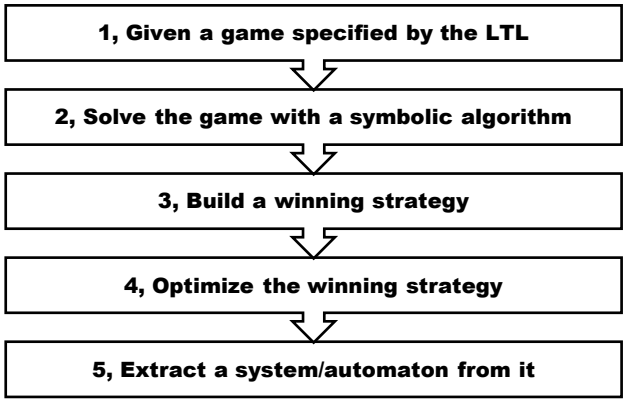
As mentioned before, the synthesis process can be viewed as a game played by system and environment. So far, all the components of the game are specified by the formulas discussed above, include initial states, transition relations, and winning condition. It's worth mentioning that the name of formula that defines the winning condition is Generalized Reactivity (1) [GR (1)] formula,

$$\Phi = (\varphi_g^e \Rightarrow \varphi_g^s)$$

The GR (1) formula Φ can be satisfied not only by φ_g^s is true, but also by φ_g^e is false. The former implicates the system meets its goal specification despite what the environment does. The latter implicates the environment didn't reach its goal and the extraction of automaton is not guaranteed.

C. *Reactive synthesis*

The chart below shows the major steps of the synthesis process.



In step 2, the game is solved by the μ-calculus formula [5]. In step 3, the extracted strategy is based on the solution of the game computed from step 2. In step 4, the optimization work is mainly reflected in reducing the size of strategy. Finally, a system can be generated from this computation in step 5.

IV. Conclusion

This review briefly gone through the basic concepts of infinite games and the application of solving that game in the reactive synthesis process. The components of the LTL specification can be mapped to the elements of the games so that the synthesis can be done under the model of the game. In addition, the details of the algorithms of solving the game in the context of specification synthesis remains further investigation.

References

[1] Erich Gradel, Wolfgang Thomas, Thomas Wilke (Eds.), "Automata, Logics, and Infinite Games A Guide to Current Research", Springer

[2] Hadas Kress-Gazit, Member, IEEE, Georgios E. Fainekos, Member, IEEE, and George J. Pappas, Fellow, IEEE, "Temporal-Logic-Based Reactive Mission and Motion Planning", IEEE TRANSACTIONS ON ROBOTICS, VOL. 25, NO. 6, DECEMBER 2009

[3] "Parity game", Wikipedia, https://en.wikipedia.org/wiki/Parity_game

[4] Tutorial: Synthesis, Seminar "Games, Synthesis, and Robotics", Bernd Finkbeiner, Universitat des Saarlandes, https://www.react.uni-saarland.de/teaching/games-synthesis-robotics-11/tutorial.pdf

[5] Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, Yaniv Sa'ar, "Synthesis of Reactive (1) designs", Journal of Computer and System Sciences

[6] "Linear temporal logic", Wikipedia, https://en.wikipedia.org/wiki/Linear_temporal_logic

[7] Zielonka, W (1998). "Infinite Games on Finitely Coloured Graphs with Applications to Automata on Infinite Trees". Theor. Comput. Sci. 200 (1–2): 135–183. doi:10.1016/S0304-3975(98)00009-7 (https://doi.org/10.1016%2FS0304-3975%2898%2900009-7).