Using Alternating-Time Logic for Modeling of Artificial Agents in Wireless Nets

TID Presentation

Ing. Jiří Král

Faculty of Information Technology

Contents

Motivation

- Branching Time Logic (e.g. CTL,CTL*) - used for Agent and Multiagent systems.
- CTL enforces universal $\forall$ or existencial $\exists$ quantificator.
- ATL offers quantification over selective paths - a generalization of CTL.
- Examples on board.
  - $\forall \bigcirc p$
  - $\langle\langle A \rangle\rangle \bigcirc p$

Alternating Time Logic

- Concurent Game Structure
- Fairness Constraints
- ATL Syntax
- ATL Semantics

Concurent Game Structures

A *concurent game structure* is a tuple

$$S = \langle k, Q, \Pi, \pi, d, \delta \rangle$$

where

- $k > 1$ is a natural number of players. Each player is identified by number $1, \ldots, k$.
- Q is a finite set of *states*.
- Π is a finite set of *propositions*.
- For each state $q \in Q$, a set $\pi(q) \subseteq \Pi$ of propositions true at q. Function π is called *labeling function*.

Concurent Game Structures - # 2

- For each player $a \in 1, \ldots, k$ and each state $q \in Q$, a natural number $d_a(q) \geq 1$ of moves avilable at state q to a player a (each move is identified by a number). For each state $q \in Q$, a *move vector* at q is a tuple $\langle j_1, \ldots, j_k \rangle$ for each player a. Given state $q \in Q$, we write $D(q)$ for the set $1, \ldots, d_1(q) \times 1, \ldots, d_k(q)$ of moves of move vectors. The function D is called *move function*.
- For each state $q \in Q$ and each move vector $\langle j_1, \ldots, j_k \rangle \in D(q)$, a state $\delta(q, j_1, \ldots, j_k) \in Q$, that results from state q if every player $a \in 1, \ldots, k$ choose move j_a. The function δ is called *transition function*.

Concurent Game Structures - # 3

- A state q' is a *Successor* of q if there is a move vector such that $q' = \delta(q, j_1, \ldots, j_k)$.
- An infinite sequence $\lambda = q_0, q_1, \ldots$ is a *Computation* of S of states such that for all positions $i \geq 0$ q_{i+1} is sucesor of q_i. A *q-computation* is a computation starting from state q. Notation $\lambda[i]$ denotes the i-th position of computation λ

Concurent Game Structures - example

- System with processes a and b. The process a assigns values to the boolean variable x. When $x = false$, then a can leave the value of x unchanged or change it to $true$. When $x = true$, then a leaves the value of x unchanged. In a similar way, the process b assigns values to y.
- Model of this system is:
  - $\Pi = x, y$
  - $\Sigma = a, b$
  - $Q = q, q_y, q_x, q_{xy}$. The state q corresponds to $x = y = false$, the state q_x corresponds to $x = true$ and $y = false$, and similarly for q_y and q_{xy}.
  - Labeling function coresponds to names of states q_{xy} means $\pi(q_{xy}) = x, y$
  - $d_1(q) = d_1(q_y) = 2$ and $d_1(q_x) = d_1 q_{xy} = 1$
  - $d_2(q) = d_2(q_x) = 2$ and $d_2(q_y) = d_2 q_{xy} = 1$
  - $\delta(q, 1, 1) = q$, $d(q, 1, 2) = q_y$ ...

Fairness Constraints

- A fairness constraint in a game structure $S = \langle k, Q, \Pi, \pi, d, \delta \rangle$ is a tuple $\langle a, \gamma \rangle$, where $a \in 1, \ldots, k$ is a player and a function γ maps every state $q \in Q$ to a subset of moves available at state q to player a.
- Consider a computation $\lambda = q_1, q_2, \ldots$ of game structure S and fairness constraint $\langle a, \gamma \rangle$. We say that $\langle a, \gamma \rangle$ is *enabled* at position $i \geq 0$ of λ if $\gamma(q_i) = \emptyset$
- We say that $\langle a, \gamma \rangle$ is *taken* at position $i \geq 0$ of λ if there is a move vector $\langle j_1, \ldots, j_k \rangle$ such that $j_a \in \gamma(q_i)$ and $\delta(q_i, j_1, \ldots, j_k) = q_{i+1}$.

ATL - Syntax

Definition with respect to: Π a finite set of propositions, Σ a finite set of players. An ATL formula is:

- p, for propositions $p \in \Pi$
- $\neg\phi$ or $\phi_1 \vee \phi_2$, where ϕ, ϕ_1, ϕ_2 are ATL formulas.
- $\langle\langle A \rangle\rangle \bigcirc \phi$, $\langle\langle A \rangle\rangle \Box \phi$ or $\langle\langle A \rangle\rangle \phi_1 \mathcal{U} \phi_2$, where $A \subseteq \Sigma$ is a set of players, ϕ, ϕ_1, ϕ_2 are ATL formulas.

The operator $\langle\langle\rangle\rangle$ is a path quantifier, $\bigcirc$ (next), $\Box$ (always) and $\mathcal{U}$ (until) are temporal operators.

ATL - Strategy

Definitions with respect to: $S = \langle k, Q, \Pi, \pi, d, \delta \rangle$.

- A *strategy* for a player $a \in \Sigma$ is a function f_a that maps every nonempty finite state sequence $\sigma \in Q^+$ to a natural number such that: $f_a(\lambda) \leq d_a(q)$.
- Given $q \in Q$, $A \subseteq 1, \ldots, k$ and a set $F_A = f_a | a \in A$ of strategies, one for each player in A, we define *outcomes* of F_A from q to be the set $out(q, F_A)$ of q-computations tht players in A enforce when follow strategies in F_A.

ATL - Semantics

We write $S, q \models \phi$ to indicate that q satisfies the formula ϕ in structure S. The definition of $\models$ is:

- $q \models p$ for propositions, iff $p \in \pi(q)$.
- $q \models \neg\phi$ iff $q \not\models \phi$.
- $q \models \phi_1 \vee \phi_2$ iff q $\models \phi_1$ or $\models \phi_2$.
- $q \models \langle\langle A \rangle\rangle \bigcirc \phi$ iff there exist set F_A of strategies, such that for all $\lambda \in out(q, F_A)$ we have $\lambda[1] = \phi$.
- $q \models \langle\langle A \rangle\rangle \Box \phi$ iff there exist set F_A of strategies, such that for all $\lambda \in out(q, F_A)$ and all positions $i \geq 0$ we have $\lambda[i] = \phi$.
- $q \models \langle\langle A \rangle\rangle \phi_1 \mathcal{U} \phi_2$ iff there exist set F_A of strategies, such that for all $\lambda \in out(q, F_A)$ there exists a position $i \geq 0$ such that $\lambda[i] = \phi_2$ and for all positions $0 \leq j \leq i$ we have $\lambda[j] = \phi_1$.

Model Checking and Complexity

- Model checking of ATL is identical to algorithm of CTL
- Exception: function *Pre* that from a set of players A and set of states ρ returns the set of states q such that from q players in A enforces the next state to lie in ρ
- Function *Pre* - highest complexity
- Comparision of closed and opened systems

	Closed	**Opened**
ATL joint complexity	PTIME	PTIME
ATL structure compexity	NLOGSPACE	PTIME
ATL* joint complexity	PSPACE	2EXPTIME

Application in Wireless Nets

- Necessity of open system representation
- Usability in wireless sensor nets
  - Battery limits
  - Complexity of computations
- Usability in wired nets
  - Usually more flexible resources

Conclusion

- ATL offers a represenation of an opened system
- ATL is more expressive than CTL (and ATL* more than CTL*)
  - Path quantifications
  - More flexible constraints
- Higher requirements for target platform

The End

Thank you for your attention.