Degree of Lookahead in Regular Infinite Games

Michael Holtmann, Lukasz Kaiser,
and Wolfgang Thomas

Infinite games

- Two players – INPUT, OUTPUT
- Set of ω words over $\sum x \sum$ (specification)
- In each round
  - Player INPUT chooses $a \in \sum$
  - Player OUTPUT responds with $b \in \sum$
- Play forms ω word $(\alpha,\beta) \in (\sum x \sum)^{\omega}$
- OUTPUT wins iff $(\alpha,\beta) \in$ Specification.

Specification formats

- ω regular language
  - ω regular expression.
  - Deterministic Parity Automaton.
  - MLO formula
- Context free ω language
- Others
  - Not in this lecture

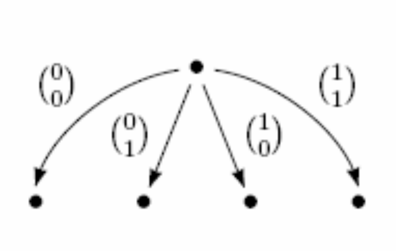
1. $\forall t(\alpha(t) = 1 \quad \rightarrow \quad \beta(t) = 1)$
2. $\neg\exists t\ \beta(t) = \beta(t+1) = 0$
3. $\exists^{\omega} t\ \alpha(t) = 0 \quad \rightarrow \quad \exists^{\omega} t\ \beta(t) = 0$

Regular language ω

- Language L is ω regular if
  - $L = U(V)^{\omega}$
    - And U, V are regular languages.
  - $L = L_1 \cup L_2$
    - And L_1, L_2 are ω regular languages.

Parity Automaton

- DPA = (Q,q_0, δ,c)
- c is coloring function c:Q→{0,…,m}
- A run of DPA:
  - For infinite word α, the set Inf(ρ_α) is the states of states visited infinitely often in run ρ_α
  - We define DPA to accept α iff max{Inf(c(ρ_α))} if even.



- Every Non deterministic parity automaton has an equivalent DPA.

ω Context free languages

- Same idea as DPA only with non deterministic push-down automata.

Game without delay

- Player INPUT chose $a \in \sum$
- Player OUTPUT immediately responds with $b \in \sum$
  - OUTPUT response depends only in previous INPUT moves.
- Büchi-Landweber Thm:
  - For a game with regular winning condition:
    - it is decidable to find the winner.
    - If there exists a winning strategy for OUTPUT it is computable by FSM.

Games with delay

- Assume $\sum=\{0,1\}$
- In each round
  - Player INPUT choose one bit.
  - Player OUTPUT can respond with either 0,1 or $\perp$ which represents – wait for next bit.
- Our goal is to decide if there exists an operator λ which is a winning strategy for OUTPUT
  - Formally: $\exists\lambda$ s.t
    - $\lambda:\sum^{\omega}\rightarrow\sum^{\omega}$
    - $\forall\alpha\in\Sigma^{\omega}$, $(\alpha, \lambda(\alpha))\in L(A)$

Types of operators

- Continues operators.
  - Each output bit of $\lambda(\alpha)$ is determined by a finite prefix of α.
    - There exists function l:
      - l:{finite bit strings}$\rightarrow$\{0,1,$\perp$\}.
      - $\lambda(\alpha)=l(\alpha_0)l(\alpha_0\alpha_1)l(\alpha_0\alpha_1\alpha_2)\ldots$ - $\perp$ are omitted
      - $l(\alpha_0)l(\alpha_0\alpha_1)l(\alpha_0\alpha_1\alpha_2)$ does not end with infinite $\perp$
- f – delay operators.
  - There exists function f:N$\rightarrow$N
    - The bit $\lambda(\alpha)_i$ depends only in $\alpha_0\alpha_1\ldots\alpha_{f(i)}$
  - For $\lambda:\Sigma^\omega\rightarrow\Sigma^\omega$ equivalent to continuous operator.
    - König's Lemma
    - Every continuous operator over bounded close space is uniformly continous
- d – delay operators.
  - Same for f(i) = i + d

Context free games with delay

- Can we decide if OUTPUT wins the game with f-delay?
  - No, since it is not decidable if $L \in CFL_\omega$ is universal.

- If OUTPUT can win with f-delay, can it always win with d-delay?
  - No. For specification:
    - INPUT=$1^{2m_0}0^{n_0}1^{2m_1}0^{n_1}\ldots$ for $m_i, n_i \in N$
    - OUTPUT = $1^{m_0}0^{m_0+n_0}1^{m_1}0^{m_1+n_1}\ldots$

Regular games with delay

- Theorem:
  – Let *A* be a DPA over $\{0,1\}^2$
  – There is a continuous operator λ s.t OUTPUT wins with λ-delay iff there exists $d \in N$ s.t OUTPUT wins with d-delay.
- Corollary:
  – It is decidable to know if OUTPUT can win with a continuous operator delay.
    - Reduction to parity game
    - 3EXP TIME complexity

Proof plan

- Notations
- Reduction to Block Game
- Reduction to Semi Group Game

Notations

- From now on f(i) stands for the number of bits that player OUTPUT can wait (output ⊥) before responding with the i_{th} bit.
  - Formally f(i) =old_f(i) – old_f(i-1)
- We assume that the DPA *A* is fixed.
- For a given function f, we mark the game as Γ_f
  - In this game OUTPUT may wait f(i) bits

The Block Game (f)

- Instead of choosing bit in every round, each player chooses word.
- First round:
  - Player INPUT chooses two words u_0 and u_1.
    - $f(0) \leq |u_0| \leq 2f(0)$, $f(1) \leq |u_1| \leq 2f(1)$
  - Player OUTPUT responds with v_0.
    - $|v_0| = |u_0|$
- Other rounds:
  - INPUT chooses u_i s.t $f(i) \leq |u_i| \leq 2f(i)$
  - OUTPUT responds with v_{i-1} s.t $|v_{i-1}| = |u_{i-1}|$

The block game – example

- Assume f = {2, 5, 9, 2…}
- Round 0:
  - INPUT : 01011011
  - OUTPUT : 110
- Round 1:
  - INPUT : 010110111110001011
  - OUTPUT : 11010010
- Round 2:
  - INPUT : 010110111110001011101
  - OUTPUT : 110100101000000001
- …….

$\Gamma \Rightarrow$ Block Game

- Proposition:
  - $\forall f, \exists g$ s.t if INPUT wins on Γ_g he wins on Block game(f)
- Corollary:
  - If INPUT wins in Γ (i.e - for every f) then INPUT wins in Block Game (for every f).

$\Gamma \Rightarrow$ Block Game: Proof

$\forall$f, $\exists$g if INPUT wins on Γ_g he wins on Block game(f)

- Proof:
  - First move:
    - Need to find two words, with length at least f(0), f(1), as player INPUT first move in Block game(f).
      - Solution: Set g(0) = f(0) + f(1), and follow INPUT first move in Γ_g.
    - OUTPUT responds with f(0) bits.
  - Next move:
    - In Γg player OUTPUT choose at least one bit, so we can simulate INPUT next move. Let g(1) = f(2). And in general g (i)=f(i+1). If OUTPUT choose more than one bit, then INPUT can choose even more than g(i) bits, but we can choose only the g(i) prefix.

Γ ⇒ Block Game: simulation

- g(0) = f(0) + f(1), g(i) = f(i+1). Assume f={2,5,3,…}
- Round 0: g(0) = 7, f(0)=2, f(1) = 5
  - Γg
    - INPUT : 1011011
    - OUTPUT : 1
  - Block game
    - INPUT : 1011011
    - OUPUT : 10
- Round 1: g(1) = 3, f(2)=3
  - Γg
    - INPUT : 1011011 + 011
    - OUTPUT : 10
  - Block game
    - INPUT : 1011011011011
    - OUPUT : 1001110001

Block game $\Rightarrow \Gamma$

- Proposition:
  - $\forall f, \exists g$ s.t if INPUT wins on Block game(g) he wins on Γ_f
- Corollary:
  - If INPUT wins in Block game (i.e - for every f) then INPUT wins in Γ (for every f).

Block game $\Rightarrow \Gamma$: Proof

$\forall f, \exists g$ s.t if INPUT wins on Block game(g) he wins on Γ_f

- Proof:
  - First move:
    - Set g(0) = f(0), and set g(1) to be long enough for OUTPUT to respond 2f(0) times.
    - We set INPUT move in Γ to be the first two chosen words in the block game.
    - OUTPUT must respond in Γ with a word with length 2f(0). So we can simulate that move in block game.
  - Next move:
    - We set g(i) to be long enough so OUTPUT must respond with 2f(i-1) bits in the Γ game, and simulate its response in the block game,

Block game $\Rightarrow \Gamma$: Simulation

- Assume f={2,2,2,2,….}
- Round 0: f(0) = 2, g(0)=2, g(1) = 8
  - Block game(g)
    - INPUT : 0111110111001
    - OUTPUT : 001
  - Γ_f
    - INPUT : 011111
    - OUPUT : 001
- Round 1: f(1) = 2, g(1)=8, g(2) = 32
  - Block game(g)
    - INPUT : 0111110111001000000111011100 0…
    - OUTPUT : 00100010
  - Γ_f
    - INPUT : 0111110111001000
    - OUPUT : 00100010
- …

Semi Group Game

- For a given DPA *A*:
  - Every two strings u,v (|u|=|v|), forms matrix μ (u,v) with size |Q|x|Q|:
    - For every states p,q the p,q cell is:
      - $-\infty$ if $\delta^*(p,(u,v)) \neq q$
      - Maximal color in the associated path from p to q

Semi Group Game – ~ equivalent class

- $(u,v) \sim (w,x) \Leftrightarrow \mu(u,v) = \mu(w,x)$
- Note:
  - Since Q and c are finite there is a finite number of equivalent classes [(u,v)]
    - Therefore at least one equivalent class has infinite size.
  - It is possible to recognize [(u,v)] via finite automaton
    - Simulate behavior of DPA on finite strings
  - One can compute all equivalent classes [u,v]

Semi Group Game – ≈ equivalent class

- u≈w ⇔ If (u,v)∈[(a,b)] for some v, then ∃x s.t (w,x)∈[(a,b)], for every a,b.
  - Intuitively:
    - OUPUT can choose same ~ equivalent class for u and w.
- Note:
  - Number of equivalent class is finite.
  - Possible to recognize [u] via finite automaton.
  - One can compute all equivalent classes.

Semi Group Game

- First round:
  - INPUT choose two infinite size classes $[u_0]$, $[u_1]$.
  - OUTPUT responds with $[(u_0,v_0)]$ with infinite size.
- Next rounds:
  - INPUT chooses $[u_i]$
  - OUTPUT responds with $[(u_{i-1},v_{i-1})]$ with infinite size.
- Winning condition:
  - $(u_0,v_0),(u_1,v_1),\ldots \in L(A)$

Block Game $\Rightarrow$ Semi Group Game

- $\exists f$ s.t $\forall g \subseteq f$, INPUT wins block game(g) $\Rightarrow$ INPUT wins semi group game.

  Lemma:
    - INPUT wins block game $\Leftrightarrow$
    - $\exists f$ s.t $\forall g \subseteq f$, INPUT wins block game(g)

- Proof
  - Let d' be the longest word in all finite classes [u], define g(i) = max{ f(i), d' }.
  - INPUT wins block game(g)
    - Apply same strategy on semi group game

Semi Group Game ⇒ Block Game

- INPUT wins semi group game ⇒ ∃f s.t ∀g ⊆ f, INPUT wins block game(g).
- Proof
  - Assume INPUT chooses [u] in the semi group game.
    - Let $A_{[u]}$ be automata recognize [u].
    - Let n' be the maximal number of states among these automata for every $u \in \Sigma^*$
    - Set f(i) = n'
    - Since [u] is infinite, $\exists w \in [u]$, $f \leq |w| \leq f + |A_{[u]}|$
      - Therefore $f \leq |w| \leq 2f$
  - It is possible to choose $w \in [u]$ in block game(f)
  - Same arguments holds for g ⊆f

Semi Group Game $\Leftrightarrow \Gamma_{<2n'-1>}$

- Theorem:
  - OUTPUT wins semi group game iff it wins Γ with constant delay of 2n'-1.
    - $\Gamma \Rightarrow$ Block Game $\Rightarrow$ Semi Group Game
    - Semi Group Game $\Rightarrow$ Block Game(n') $\Rightarrow \Gamma_{<2n'-1>}$
- Corollary:
  - OUTPUT wins Γ with finite delay iff it wins Γ with 2n'-1 delay.
- $n' \leq 2^{(mn)^{2n}}$

Open questions

- Infinite delay
  - OUTPUT may request information on infinite number of INPUT bits (for example – all even bits)
- ω context free languages – too wide
  - Deterministic ω context free specifications
    - Decidable?
    - Conjecture:
      - Polynomial delay is enough. i.e $f(i) = poly(i)$