More on Complexity

Natasha Alechina   Brian Logan

Utrecht University

n.a.alechina@uu.nl   b.s.logan@uu.nl

Time Complexity

- suppose the input to an algorithm has size n (for example, n is the number of variables in a propositional formula)
- suppose on any input, the algorithm always makes the same number of steps, c, regardless of how large n is (for example, it prints c 0s to the screen...). Then the growth rate of the running time as a function of input size is $O(1)$: constant time
- suppose the algorithm takes at most $c_1 \times n + c_2$ steps, where c_1 and c_2 are constants. This is $O(n)$: linear time

Time Complexity continued

- suppose the algorithm generates all triples of variables and then all pairs and iterates over them c_1 times: $c_1 \times n^3 + c_1 \times n^2$. This is $O(n^3)$: polynomial time (generally, $O(n^c)$).
- suppose the algorithm generates and checks all assignments of 0 and 1 to the n variables. There are 2^n assignments, and suppose checking each takes n steps. Then the algorithms takes at most $n \times 2^n$ steps. This is $O(2^n)$: exponential time.
- generally, exponential time is $O(2^{polynomial(n)})$, or $O(2^{n^c})$

Size of the input

- usually the size of the input is characterised by several parameters
- for example, the number of agents k, the number of states n, the number of actions d
- the definition of time complexity is the same
- $O(n^2 \times k \times d)$ is polynomial in the input size
- $O(n \times d^k)$ is exponential (in the number of agents k).

Deterministic time complexity classes

- P, or PTIME, is the set of decision problems that can be solved in time $O(n^c)$, for some constant c
- EXPTIME is the set of decision problems that can be solved in time $O(2^{n^c})$, for some constant c

A closer look at the complexity of ATL model checking

- PTIME complexity result for ATL model checking is relative to the size of the model and the formula
- size of the model is understood as the explicit representation, listing every state and every transition
- the number of states can be exponential in the number of propositional variables
- for $|PV|$ variables, we have $2^{|PV|}$ number of states
- ISPL encoding may have one Evolution clause for each of $|PV|$ variables, but the transition function has to be specified for each of $2^{|PV|}$ states

A closer look at the complexity of ATL model checking

- PTIME complexity result for ATL model checking is relative to the size of the model and the formula
- size of the model is understood as the explicit representation, listing every state and every transition
- the number of transitions can be exponential in the number of agents
- m: transitions, n: states, d: actions k: agents
- $m = O(nd^k)$ (from each state there may be d^k transitions)
- ATL model checking is $\Delta_3^{\mathbf{P}}$-complete with respect to the number of states, agents, actions, and implicit transitions (next slide)
- $\Delta_3^{\mathbf{P}} = \mathbf{P}^{\mathbf{NP}^{\mathbf{NP}}}$ is worse than $\mathbf{P}^{\mathbf{NP}}$ which is worse than NP (it assumes an NP oracle that can ask another NP oracle)

Implicit representations of transitions

- implicit concurrent game models: (similar to ISPL)
- implicit transition function $\hat{o}$
- for each state q^r, define an ordered set of pairs $(\varphi_0^r, q_0^r), \ldots, (\varphi_{t_r}^r, q_{t_r}^r)$, where each φ_i^r is a formula specifying a set of actions that may be executed in q^r, and q_i^r is the resulting state
- φ_i^r is a boolean combination of statements $Agent_1.Action = \alpha$
- $\varphi_{t_r}^r$ is $\top$
- the first φ_i^r formula that holds determines the resulting state q_i^r

Example of implicit representation

- for example, q is the initial state, there are k agents, and each agent has two actions α and β (2^k joint actions)
- if all agents execute action α, the system goes into $q\alpha$ state
- otherwise the system goes into $q\beta$ state
- in $q\alpha$ and $q\beta$, any action loops back to the same state
- from q:
  - $(Agent_1.Action = \alpha \wedge \ldots \wedge Agent_k.Action = \alpha, q\alpha)$,
  - $(Agent_1.Action = \beta \vee \ldots \vee Agent_k.Action = \beta), q\beta)$,
  - $(\top, q)$
- from $q\alpha$: $(\top : q_\alpha)$
- from $q\beta$: $(\top : q_\beta)$

Complexity is sensitive to how we measure input size

- complexity is very sensitive to the context!
- in particular, the way we define the parameters, and measure their size, is crucial