Quantum Finite Automata and Logics

Logic and classical automata

- Büchi theorem
    - A language of finite words is recognizable by a finite automaton iff it is monadic second-order definable, and both conversions, from automata to formulas and vice versa, are effective.

Monadic second order logic (MSO)

- Input word - $a_1 a_2 a_3 \ldots a_n \quad (a_i \in \{a, b\})$

- Predicates:

  - $Q_{b_i}(x) = \{\text{the symbol of number } x \text{ in the input word equals } b_i\} \quad (b_i \in \{a, b\})$
  - $S(x, y) = \{y = x + 1\}$
  - $first(x) = \{x = 1\}$
  - $last(x) = \{x = n\}$

Monadic second order logic 2

- The length of the input word is a multiple of 3.
- The set variables:
  - $X_1 = \{\text{all the positions } i \text{ such that } i \equiv 1 \,(mod\, 3)\}$
  - $X_2 = \{\text{all the positions } i \text{ such that } i \equiv 2 \,(mod\, 3)\}$
  - $X_0 = \{\text{all the positions } i \text{ such that } i \equiv 0 \,(mod\, 3)\}$

Monadic second order logic 3

$$\exists X_1 X_2 X_0((X_1 \cap X_2 = \emptyset) \wedge (X_1 \cap X_0 = \emptyset) \wedge (X_2 \cap X_1 = \emptyset) \wedge$$
$$\wedge \forall x(\mathit{first}(x) \Rightarrow X_1(x)) \wedge \forall xy(S(x,y) \wedge ((X_1(x) \wedge X_2(y)) \vee (X_2(x) \wedge X_0(y)) \vee (X_0(x) \wedge X_1(y)))) \wedge$$
$$\wedge \forall x(\mathit{last}(x) \Rightarrow X_0(x)))$$

Logic and classical automata 2

- Equivalence between finite automata and monadic second-order logics over infinite words and trees.
- Temporal logics.

Logic, automata, complexity

- 1974 - Fagin gave a characterization of NP as the set of properties expressible in second order existential logic.
- Complexity classes – P, PSPACE can also be characterized by mathematical logic.

Quantum finite automata

- Different notations of quantum finite automata
  - Measure once quantum finite automata
  - Measure many quantum finite automata
  - Two way quantum finite automata
  - Pushdown quantum finite automata
  - Quantum finite automata with mixed states

Measure once quantum finite automaton

- A measure once quantum finite automaton (MO-QFA) is a tuple $A=(Q,\Sigma,\delta,q_0,Q_{acc},Q_{rej})$, where Q is finite set of states, Σ is an input alphabet, $q\in Q$ is a initial state, $Q_{acc}\subseteq Q$ and $Q_{rej}\subseteq Q$ are sets of accepting and rejecting states $(Q_{acc}\cap Q_{rej}=\varnothing)$, and is the transaction function $\delta:Q\times\Gamma\times Q\rightarrow C_{[0,1]}$, where $\Gamma=\Sigma\cup\{\#,\$\}$ is working alphabet of A, and # and $ are the left and right end makers.

Measure once quantum finite automaton 2

The computation of A is performed in the inner-product space $l_2(Q)$ (with the basis $\{|q>|q\in Q\}$) using the linear operators $V_\sigma, \sigma\in\Gamma$, defined by $V_\sigma(|q>)=\sum_{q'\in Q}\delta(q,\sigma,q')|q>$ which are required to be unitary.

First order logic

- Input word $w_1 w_2 \ldots w_n$ in the finite alphabet $\Sigma = \{a_1 a_2 \ldots a_l\}$
- Predicates:
  - $Q_{a_i}(x) = \{\text{the symbol of number } x \text{ in the input word equals } a_i\} (a_i \in \Sigma)$
  - $S(x, y) = \{y = x + 1\}$, $\mathit{first}(x) = \{x = 1\}$, $\mathit{last}(x) = \{x = n\}$
- Operators - $\neg, \vee, \wedge, \Rightarrow$
- First order quantifiers - $\forall, \exists$

$$\forall x (\mathit{first}(x) \Rightarrow Q_a(x))$$

First order definable languages

- For a language $L \in \Sigma^{*}$ the following are equivalent
  - L is star-free.
  - L is recognizable by a finite aperiodic monoid.
  - L is defined by a first order formula.

First order logics and MO-QFA

- If a language in the alphabet Σ can be recognized by a measure once quantum automata and the language is first order definable, then it is an empty language or Σ^*.

Modular logic

- Input word $w_1 w_2 \dots w_n$ in the finite alphabet $\Sigma = \{a_1 a_2 \dots a_l\}$
- Predicates:
  - $Q_{a_i}(x) = \{\text{the symbol of number } x \text{ in the input word equals } a_i\} (a_i \in \Sigma)$
  - $S(x, y) = \{y = x + 1\}$, $\mathit{first}(x) = \{x = 1\}$, $\mathit{last}(x) = \{x = n\}$
- Operators - $\neg, \vee, \wedge, \Rightarrow$
- Modular quantifier $\exists^{m,n} x \phi(x)$, that means $\phi(x)$ is true for a number of x equal to n mod m.

$$\exists^{3,0} x (Q_a(x))$$

Modular logics and MO-QFA

- There exists a language that can be recognized by measure once quantum finite automata, but cannot be defined by modular logic.
- The language defined by modular formula

$$\exists^{2,1} xy (Q_a(x) \wedge S(x, y) \wedge Q_b(y))$$

cannot be recognized by a measure once quantum finite automaton.

Lindstrom quantifiers

Consider a language L over alphabet $\Sigma=(a_1, a_2, \ldots, a_s)$. Let $\bar{x}$ be a k - tuple of variables (each ranging from 1 to the input length n). In the following, we assume the lexical ordering on $\{1,2,\ldots,n\}^k$, and we write $X_1, X_2, \ldots, X_{n^k}$ for this sequence of the potential values taken on by $\bar{x}$. Let $\phi_1(\bar{x}), \phi_2(\bar{x}), \ldots, \phi_{s-1}(\bar{x})$ be s-1 Γ - formulas for some alphabet Γ . The $Q_L \bar{x}(\phi_1(\bar{x}), \phi_2(\bar{x}), \ldots, \phi_{s-1}(\bar{x}))$ holds on the string $\omega=\omega_1\omega_2\ldots\omega_n$ iff the word of length n^k whose i-th letter $(1 \leqslant i \leqslant n^k)$ is

$$\left\{\begin{array}{l} a_1, if\ \omega \models \phi_1(X_i) \\ a_2, if\ \omega \models \neg\phi_1(X_i) \wedge \phi_2(X_i) \\ a_3, if\ \omega \models \neg\phi_1(X_i) \wedge \neg\phi_2(X_i) \wedge \phi_3(X_i) \\ \ldots \\ a_s, if\ \omega \models \neg\phi_1(X_i) \wedge \neg\phi_2(X_i) \wedge \ldots \wedge \neg\phi_{s-1}(X_i) \end{array}\right\}$$

belongs to L.

Languages defined using Lindstrom quantifiers

- The alphabet $\Sigma = \{0,1\}$
- The language L is defined by regular expression $(0,1)^* 0 (0,1)^*$
- $Q_L(Q_a(x))$
- bbbbbab
- 1111101

Lindstrom quantifiers and MO-QFA

- A language can be recognized by a measure once quantum finite automaton if and only if the language can be described by Lindstrom quantifier formula corresponding to group languages (languages described by deterministic finite reversible automata) using atomic formulas $Q_a(x)$.

Thank you!