CS 228 Tutorial 4

This time:
Do enough to prove
MSO = Regular languages over words

Q1 MSO_0 vs MSO

$Sing(X)$	$x = y$
$X \subseteq Y$	$x < y$
$X < Y$	$S(x, y)$
$S(X, Y)$	$X(x)$
$Q_a(X)$	$Q_a(x)$
$\forall X \varphi$	$\forall x \varphi$

$\forall x\, \varphi$

$\exists X \varphi \equiv \neg \forall X \neg \varphi$

$MSO_0 \subseteq MSO$ (Atomic)

$$Sing(X) \equiv \exists x.\ X(x) \wedge \forall y\ X(y) \Rightarrow x = y$$

$$X \subseteq Y \equiv \forall x\ X(x) \Rightarrow Y(y)$$

$$X < Y \equiv \exists x, y.\ X(x) \wedge Y(y) \wedge \forall z\ X(z) \Rightarrow z = x \wedge \forall z\ Y(z) \Rightarrow z = y \wedge x < y$$

Similarly, $S(X, Y)$

$$Q_a(X) \equiv \forall x.\ X(x) \Rightarrow Q_a(x)$$

Logical connectives

No change required

Quantifiers

Variables of MSO_0
$\subset$ Vars of MSO

$MSO \subseteq MSO_0$

$x \longmapsto X$
and put $Sing(X)$
in the clause

$x = y \longmapsto Sing(X) \wedge Sing(Y)$
$\wedge X \subseteq Y \wedge Y \subseteq X$

$x < y \mapsto Sing(X) \wedge Sing(Y)$
$\wedge X < Y$

and so on.

Boolean connectives fine
Quantifiers over SO vars fine

$\forall x \varphi \longmapsto \forall X. Sing(X) \Rightarrow \varphi[X/x]$

$\exists x \varphi \longmapsto \exists X. Sing(X) \wedge \varphi[X/x]$

replace x

Big X
to get something
in MSO

Imo, its more convenient to think of logic to automata while working with MSO

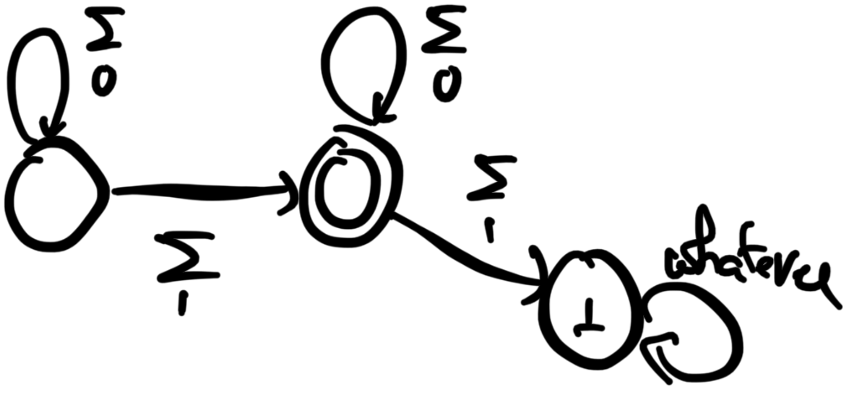
$\varphi(X_1, \dots X_n)$ over words in Σ^*

automaton alphabet is

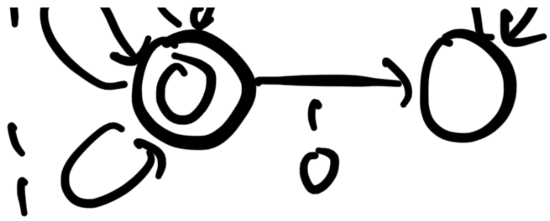
$\Sigma \times \{0,1\}^n$

signify membership of position x in X_i

Sing(X)



$X \subseteq Y$



can straightforward construct for other atomic formulae

Logical connective: Product construction!

$\wedge$ - intersection of lang

$\vee$ - union

$\neg$ - complementation

Quantifiers

How systematically deal with?

WLOG will deal with $\exists$

$$\exists x \ \underbrace{\varphi(x)}$$

$$A: (Q, \Sigma' \times \{0,1\}, \delta, q_0, F)$$

DFA

indicates membership in X

$A' : (Q, \Sigma', \delta', q_0, F)$

NFA

$\delta'(q, \alpha \in \Sigma') = \{\delta(q, (\alpha, 0)), \delta(q, (\alpha, 1))\}$

& convert this to DFA, with possibly exp. blowup.

MSO $\rightarrow$ Automaton $\equiv$ Reg lang.

Q2.

$\exists x \, \forall y \; x < y \rightarrow Q_a(y)$

Can use the discussion above. I'm lazy

$\equiv \; \exists X. \, Sing(X)$

In the remaining part, we go

Automaton $\rightarrow$ MSO

In particular $\rightarrow$ MSO_0

Principle (Proof Sketch)

If have DFA $A: (Q, \Sigma, \delta, q_0, F)$

Encode as MSO_0 over words as follows.

i) SO-variables: one for each $q \in Q$
At position x, I am in state q of A.

ii) With clauses, encode δ Key

$$\delta(q,a) = q'$$

$$\forall x,y.\ S(x,y) \wedge X_q(x) \wedge Q_a(x) \Rightarrow X_{q'}(y)$$

iii) The SO variables partition the domain.

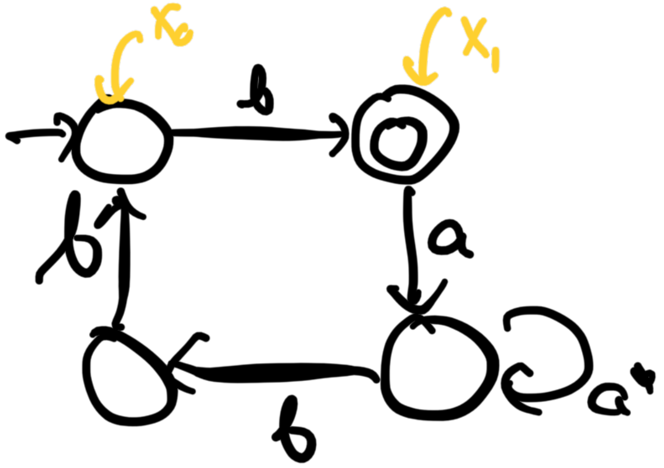
iv) $X_{q_0}(\text{first})$.

v) the last transition leads into a final state

Acceptance via DFA is something

easily doable (can encode)]
with an MSO sentence.

3)



$X_0 \xrightarrow{b} X_1$

$X_1 \xrightarrow{aa^*bb} X_0$

$\exists X_0, X_1$

$\forall x$. at most one of $X_0(x)$, $X_1(x)$

$X_0(\text{first})$

$X_0(\text{last}) \wedge Q_b(\text{last})$

$\forall x,y.\ S(x,y) \wedge X_0(x) \wedge Q_b(x) \Rightarrow X_1(y)$

$\forall y\ X_1(y) \Rightarrow \exists x.\ S(x,y) \wedge X_0(x) \wedge Q_b(x)$

$\forall x,y\ .\ x<y \wedge X_1(x) \wedge \forall z.\ x<z<y$
(neither $X_0(z)$ nor $X_1(z)$)

$$
* \left\{
\begin{array}{l}
\wedge\ \exists y_1, y_2.\ S(y_2, y_1) \wedge S(y_1, y) \\
\qquad \wedge\ Q_b(y_2) \wedge Q_b(y_1) \\
\wedge\ \forall z.\ (x < z < y_2)\ Q_a(z) \\
\wedge\ \exists z.\ x < z < y_2 \\
\Rightarrow\ X_0(y)
\end{array}
\right.
$$

$$\forall y\ X_0(y) \Rightarrow y = first$$

$$\vee$$

$$\exists x.\ *$$

i.e. justification for $X_1 \to X_0$

If I am in X_0, then I am either beginning, or I was in X_1, and saw aa^*b

Imp X_0 and X_1 ALTERNATE

Exploiting this.

—x—

Obs.

Automaton $\to$ MSO formula

formula is of the form

$\exists X_1, X_2 \dots X_n \underbrace{\varphi}$

FO formula.
i.e. no new SO vars.

—— x ——

Q4

MSO: $\varphi(X_1, \dots X_n)$ Σ

EMSO: $\exists Y_1, \dots Y_m \, \varphi(Y_1 \dots Y_m, X_1, \dots X_n)$

no quantifiers over SO variables

Corollary to logic-Automata equivalence:
every MSO over words has an equivalent EMSO formula.

Key: EMSO formulae come from automata!

1) Get automaton

$A: \langle Q, \Sigma \times \{0,1\}^n, \delta, q_0, F \rangle$

[illegible]

- If you find convenient, convert MSO to MSO_0 first

2) From automaton, get EMSO

For each $q \in Q$, we will have Y_q