



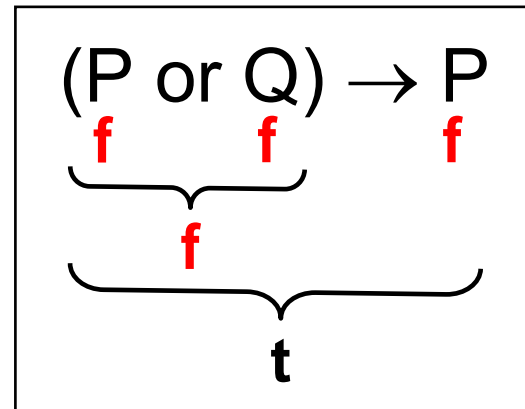
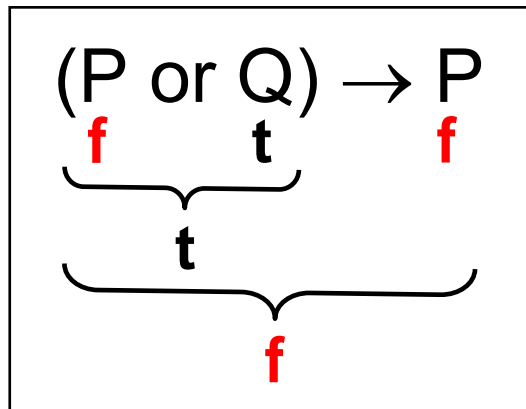
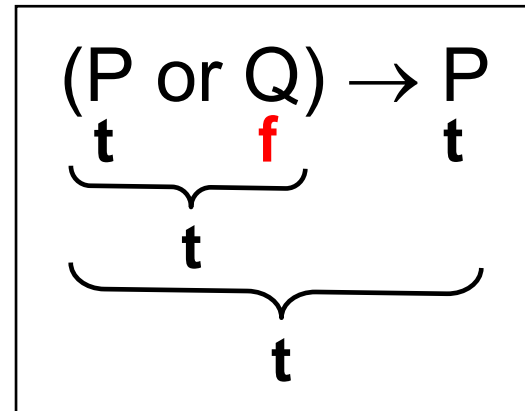
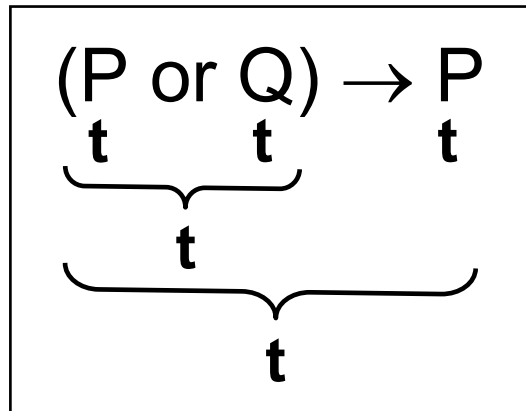
Review of Informatics 1 Computation & Logic

Basics for the exam

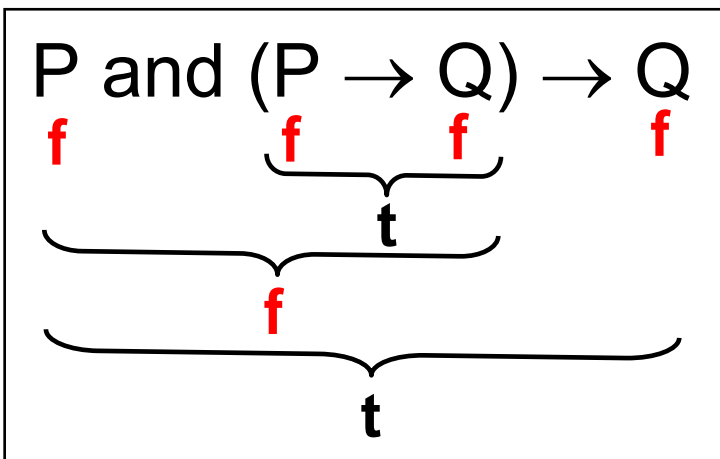
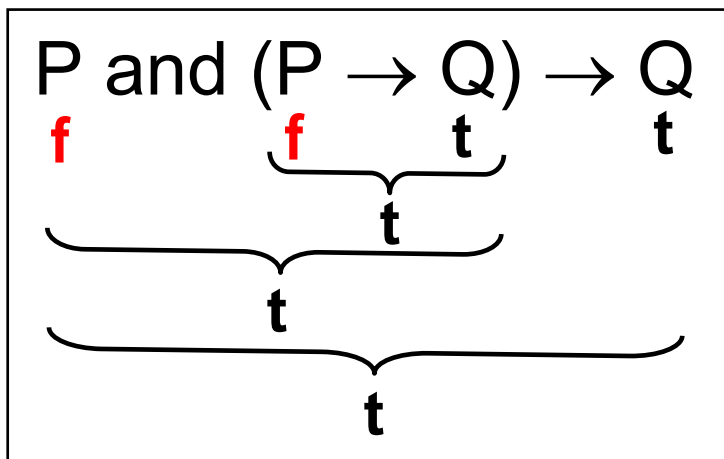
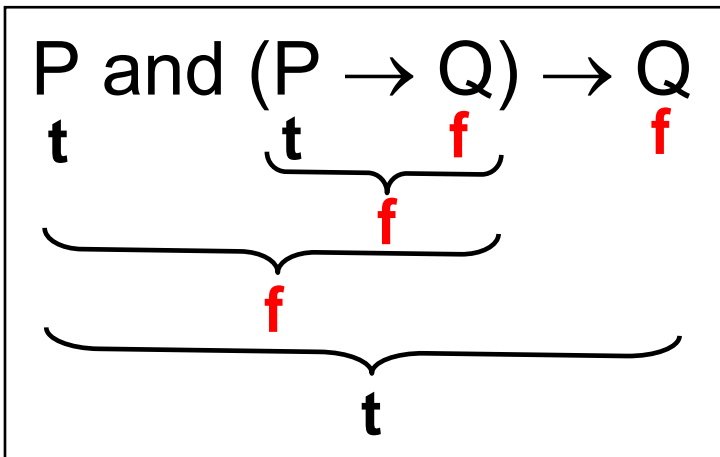
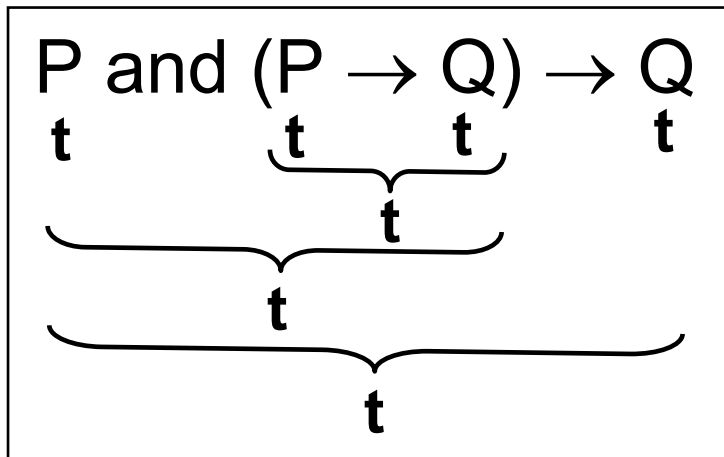
Truth Tables

P	Q	not(P)	P and Q	P or Q	$P \rightarrow Q$	$P \leftrightarrow Q$
t	t	f	t	t	t	t
t	f	f	f	t	f	f
f	t	t	f	t	t	f
f	f	t	f	f	t	t

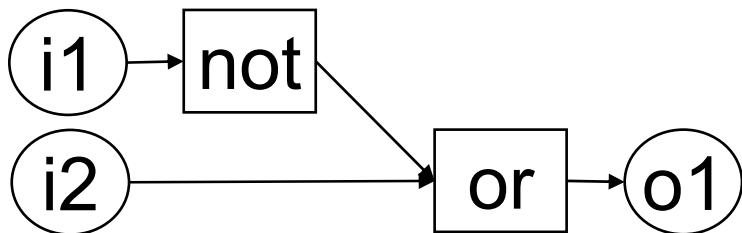
Contingencies



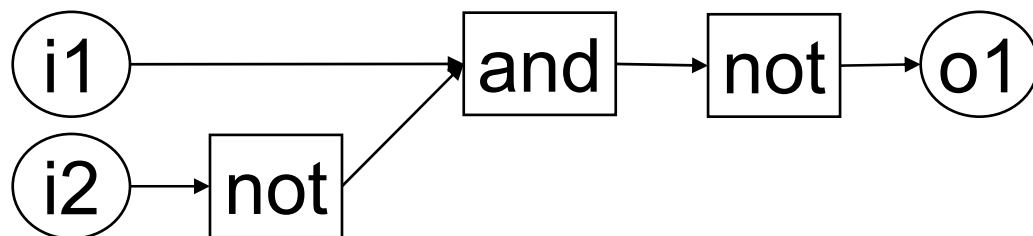
Tautologies



Typical Problem



equivalent ?



$\text{not}(i1) \text{ or } i2$

equivalent ?

$\text{not}(i1 \text{ and } \text{not}(i2))$

Proof Rules

Proof	Sub-proofs
$F \vdash A$	$A \in F$
$F \vdash A \text{ and } B$	$F \vdash A$ $F \vdash B$
$F \vdash A \text{ or } B$	$F \vdash A$
$F \vdash A \text{ or } B$	$F \vdash B$
$F \vdash C$	$A \text{ or } B \in F$ $[A F] \vdash C$ $[B F] \vdash C$
$F \vdash B$	$A \rightarrow B \in F$ $F \vdash A$
$F \vdash A \rightarrow B$	$[A F] \vdash B$

Proof Rules for Negation

Proof	Sub-proofs
$F \vdash A$	$F \vdash \text{false}$
$F \vdash \text{not}(A)$	$[A/F] \vdash \text{false}$
$F \vdash B$	$\text{not}(A) \in F \quad \vdots \quad F \vdash A$
$F \vdash A$	$F \vdash \text{not}(\text{not}(A))$

The purpose of these rules is to provide positive evidence that an expression is false.

Negation as Failure

Replace all the previous negation rules with:

Proof	Sub-proofs
$F \vdash \text{not}(A)$	$F \not\vdash A$

Where $F \not\vdash A$ means a proof can't be found for A from F

This makes the closed world assumption that F contains all the axioms pertinent to the problem and that the proof search is complete.

Normal Forms

It is not strictly necessary to use all the logical operators. For example:

$A \rightarrow B$ is equivalent to $\text{not}(A) \text{ or } B$
 $A \rightarrow B$ is equivalent to $\text{not}(A \text{ and } \text{not}(B))$
 $A \text{ or } B$ is equivalent to $\text{not}(\text{not}(A) \text{ and } \text{not}(B))$
and so on....

We can use this to convert any expression into
An equivalent one with fewer operator types.

Conversion to Clausal Form

$(a \text{ and } \text{not}(b) \rightarrow c) \text{ and } a \text{ and } \text{not}(c)$

$P \rightarrow Q$ equivalent to $\text{not}(P) \text{ or } Q$

$(\text{not}(a \text{ and } \text{not}(b)) \text{ or } c) \text{ and } a \text{ and } \text{not}(c)$

$\text{not}(P \text{ and } Q)$ equivalent to $\text{not}(P) \text{ or } \text{not}(Q)$

$(\text{not}(a) \text{ or } \text{not}(\text{not}(b)) \text{ or } c) \text{ and } a \text{ and } \text{not}(c)$

$\text{not}(\text{not}(P))$ equivalent to P

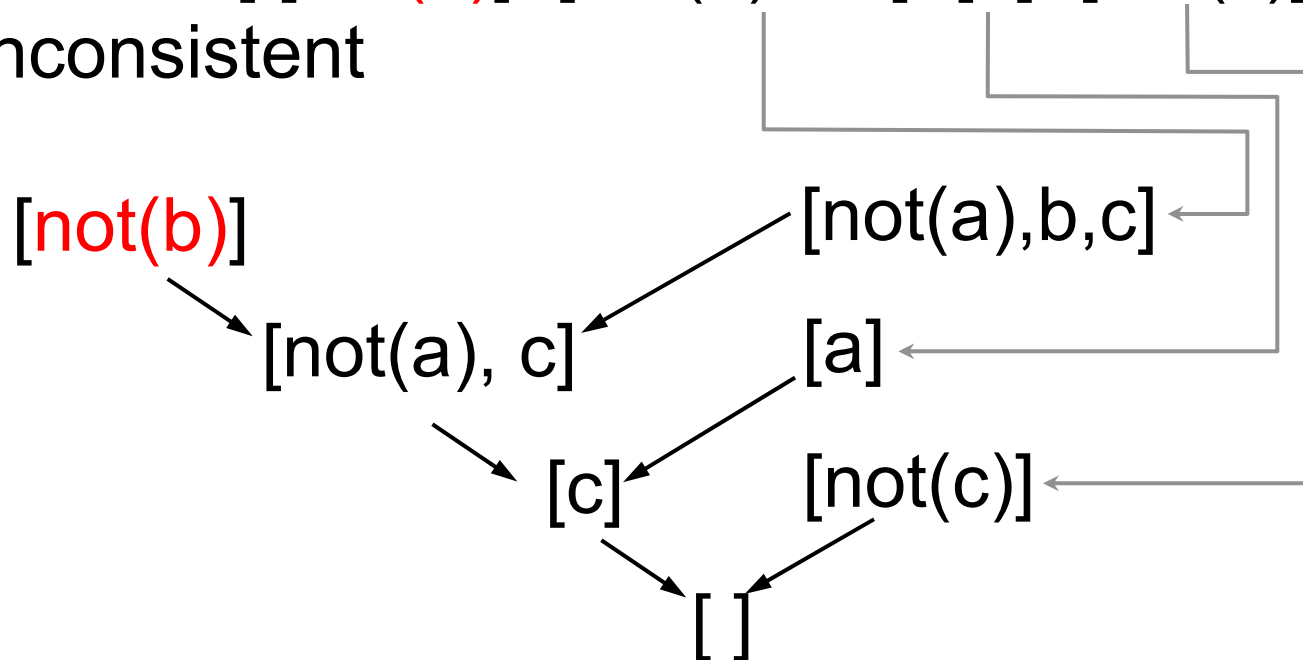
$(\text{not}(a) \text{ or } b \text{ or } c) \text{ and } a \text{ and } \text{not}(c)$

$(P \text{ or } \dots) \text{ and } \dots$ to $[[P, \dots], \dots]$

$[[\text{not}(a), b, c], [a], [\text{not}(c)]]$

A Resolution Proof

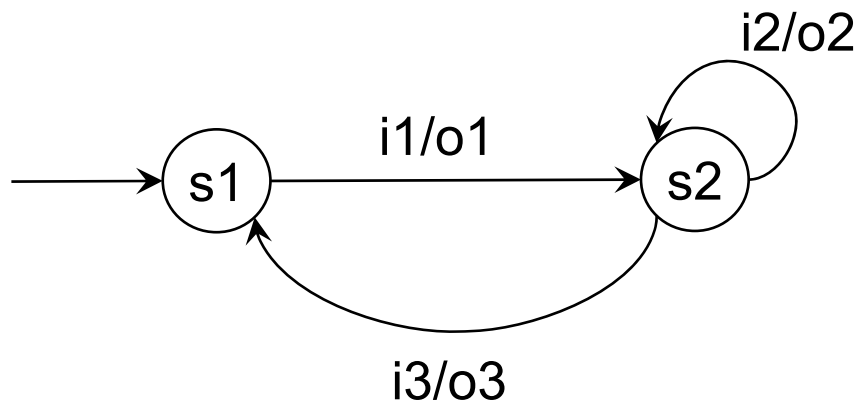
To prove b from $[\text{not}(a), b, c], [a], [\text{not}(c)]$
show that $[\text{not}(b)], [\text{not}(a), b, c], [a], [\text{not}(c)]$
is inconsistent



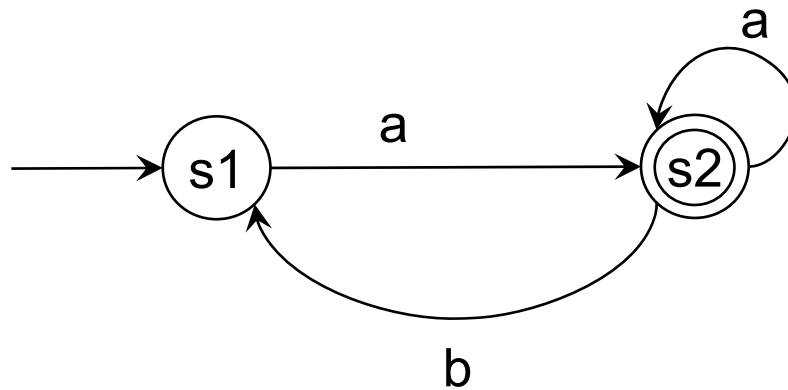
Temporal Proof Rules

Proof	Sub-proofs
$(S, J) \vdash A$	$\text{access}(J, S, F) \quad F \vdash A$
$(S, J) \vdash A \text{ and } B$	$(S, J) \vdash A \quad (S, J) \vdash B$
$(S, J) \vdash A \text{ or } B$	$(S, J) \vdash A$
$(S, J) \vdash A \text{ or } B$	$(S, J) \vdash B$
$(S, J) \vdash \text{not}(A)$	$(S, J) \not\vdash A$
$(S, J) \vdash \text{next}(A)$	$(S, J+1) \vdash A$
$(S, J) \vdash \text{prev}(A)$	$(S, J-1) \vdash A$
$(S, J) \vdash \text{e_future}(A)$	$(S, K) \vdash A \text{ for some } K > J$
$(S, J) \vdash \text{e_past}(A)$	$(S, K) \vdash B \text{ for some } K < J$
$(S, J) \vdash \text{a_future}(A)$	$(S, K) \vdash A \text{ for all } K > J$
$(S, J) \vdash \text{a_past}(A)$	$(S, K) \vdash A \text{ for all } K < J$

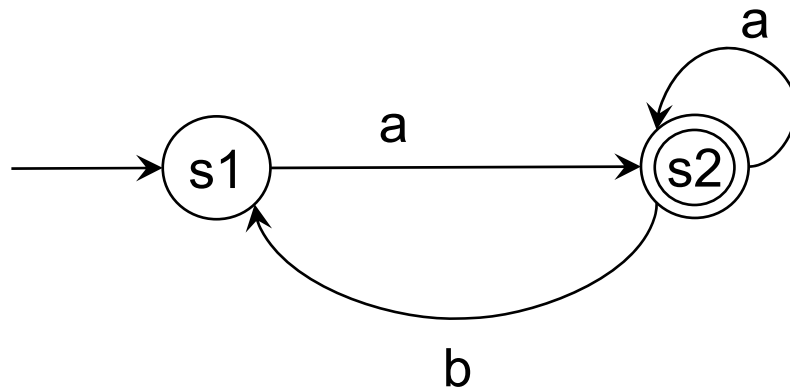
Transducer FSMs



Acceptor FSMs

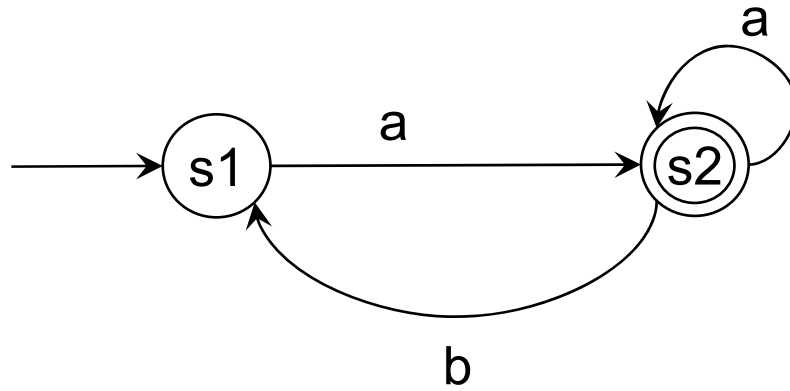


Traces



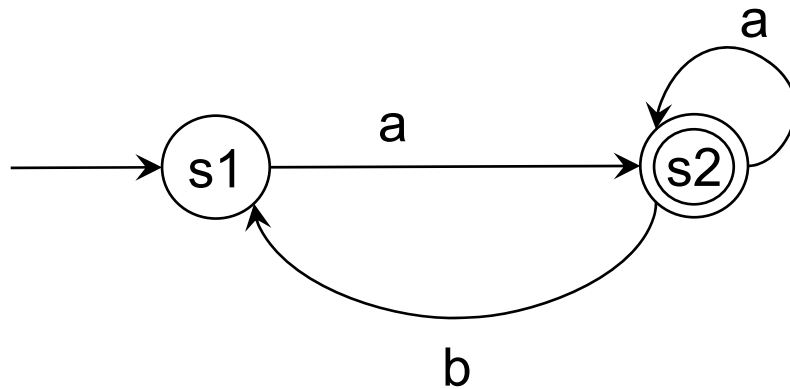
s1, a, s2, a, s2, b, s1, a, s2

Transition Function

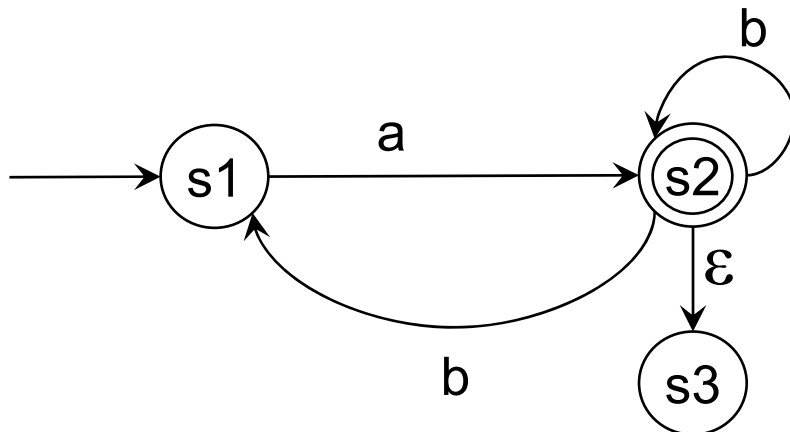


	a	b
s1	s2	-
s2	s2	s1

Deterministic v Nondeterministic

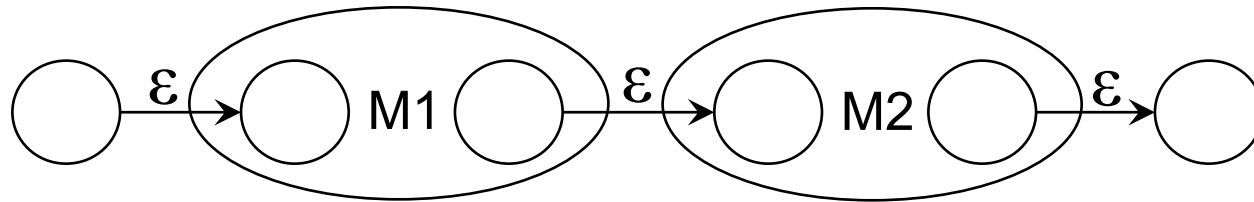


Deterministic

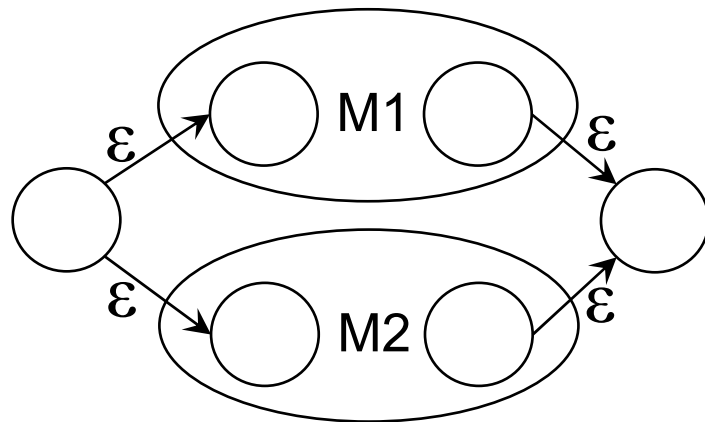


Nondeterministic

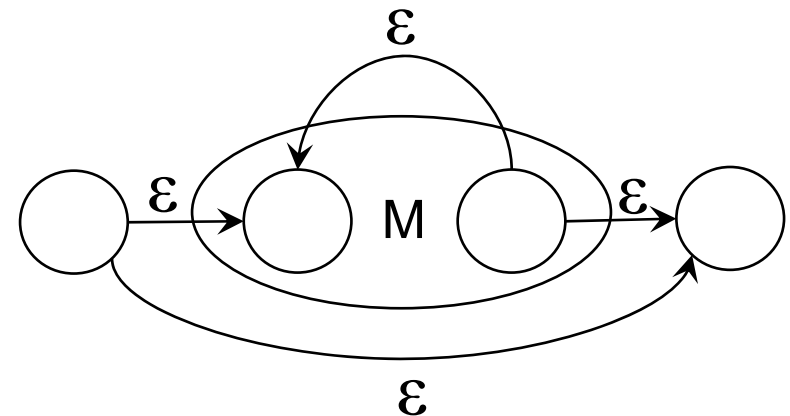
Structured FSM Design



sequence



choice



repetition

Regular Expressions

E_1E_2 Sequence

$E_1|E_2$ Choice

E^* Repetition

Half of Kleene's Theorem



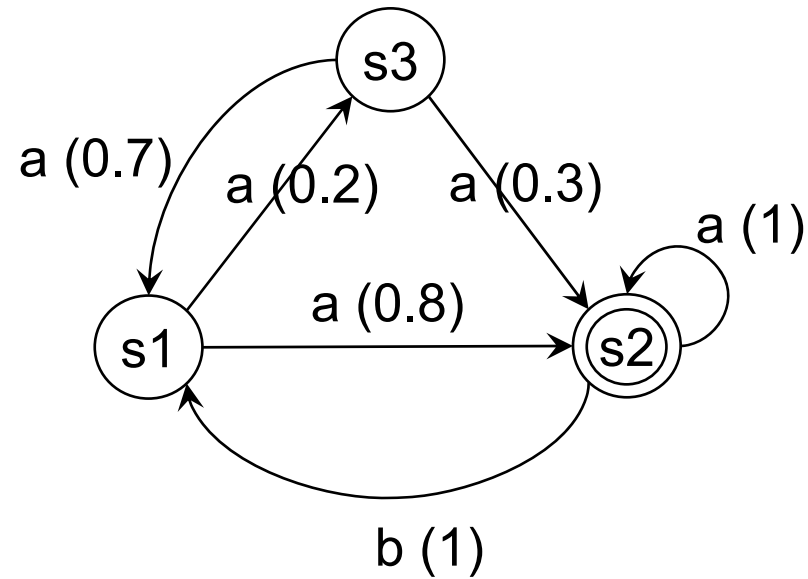
For every regular expression we can build a FSM to accept the language defined by it.

Limits of FSMs and Regular Expressions



Some languages can't be defined by FSMs or regular expressions - languages that require us to count up to arbitrarily high numbers for example.

Probabilistic FSMs





Thanks

and good luck with the exam