Undecidability of first-order logic

Ana Sokolova UNIVERSITY of SALZBURG

University of Vienna 9.10.18

www.cs.uni-salzburg.at/~anas/Undec-FOL.html

Decidability

study motivated by Hilbert's problems, in particular by the question of decidability of FOL

Alonzo Church
[1903-1995]
1935-36

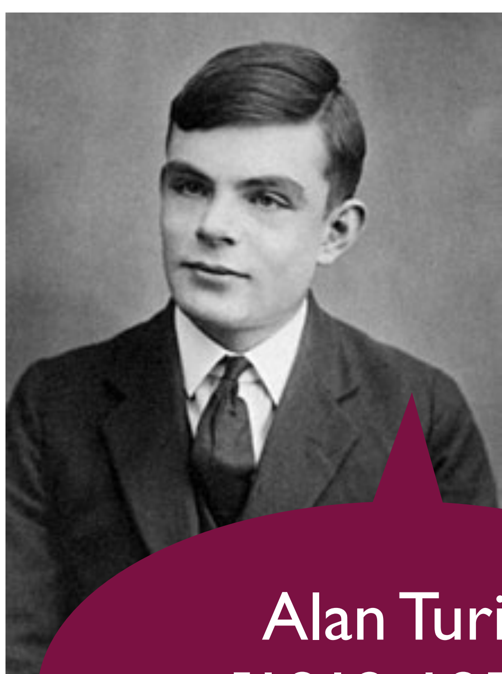
Decidability

study motivated by Hilbert's problems, in particular by the question of decidability of FOL

Alonzo Church
[1903-1995]
1935-36

Decidability

study motivated by Hilbert's problems, in particular by the question of decidability of FOL



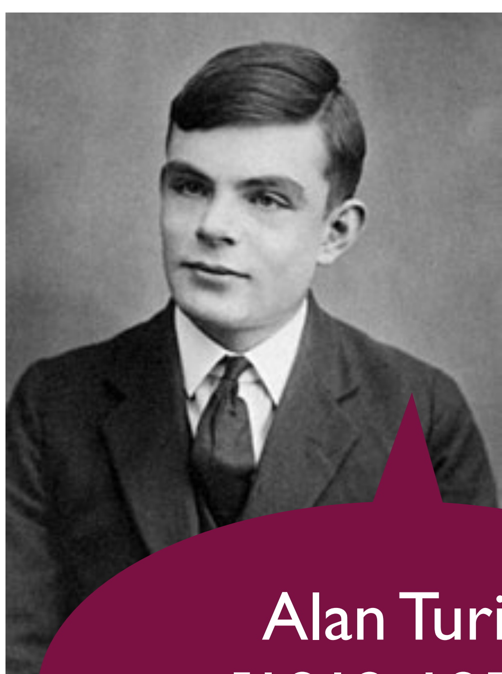
Alan Turing
[1912-1954]
1936-37

Alonzo Church
[1903-1995]
1935-36

Decidability

study motivated by Hilbert's problems, in particular by the question of decidability of FOL

A problem (set) P is decidable if there exists an algorithm that returns YES on any p ∈ P and NO otherwise.



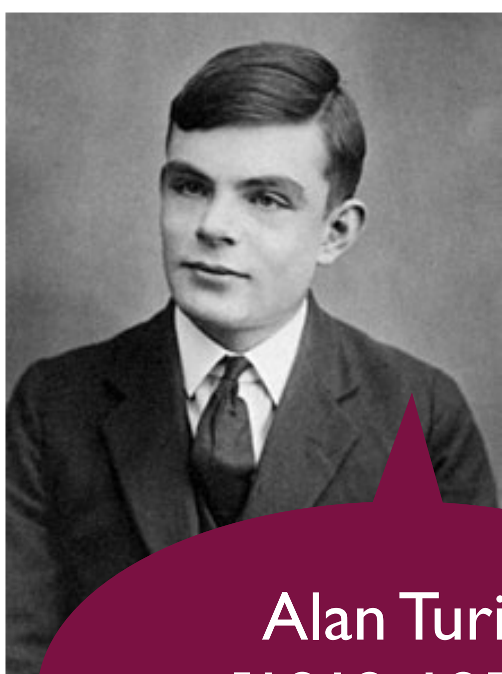
Alan Turing
[1912-1954]
1936-37

Alonzo Church
[1903-1995]
1935-36

Decidability

study motivated by Hilbert's problems, in particular by the question of decidability of FOL

A problem (set) P is decidable if there exists an algorithm that returns YES on any $p \in P$ and NO otherwise.



Alan Turing
[1912-1954]
1936-37

decision procedure
= YES/NO oracle
= computable function
= Turing machine

Hilbert's Entscheidungsproblem

1928

Is there an algorithm that decides whether arbitrary FOL formula is valid / satisfiable ?

Hilbert's Entscheidungsproblem

1928

Is there an algorithm that decides whether arbitrary FOL formula is valid / satisfiable ?

Is VALID = {φ | φ is a valid first-order formula} decidable ?

Hilbert's Entscheidungsproblem

1928

Is there an algorithm that decides whether arbitrary FOL formula is valid / satisfiable ?

Is VALID = $\{\varphi \mid \varphi$ is a valid first-order formula$\}$ decidable ?

Clearly, VALID is decidable iff
SAT = $\{\varphi \mid \varphi$ is a satisfiable first-order formula$\}$ is.

Hilbert's Entscheidungsproblem

1928

Is there an algorithm that decides whether arbitrary FOL formula is valid / satisfiable ?

Is VALID = {φ | φ is a valid first-order formula} decidable ?

Clearly, VALID is decidable iff
SAT = {φ | φ is a satisfiable first-order formula} is.

The answer is NO

Outline of Turing's proof

Reduction from the halting problem to unsatisfiability

Outline of Turing's proof

Reduction from the halting problem to unsatisfiability

Julius Richard Büchi
[1924 -1984]
1961

Lecture notes by
Larry Moss and Guram Bezhanishvili 2008

Outline of Turing's proof

Turing introduced the Turing machines and proved first:

Reduction from the halting problem to unsatisfiability

Julius Richard Büchi
[1924 -1984]
1961

Lecture notes by Larry Moss and Guram Bezhanishvili 2008

Outline of Turing's proof

Julius Richard Büchi
[1924 -1984]
1961

Turing introduced the Turing machines and proved first:

Reduction from the halting problem to unsatisfiability

Lecture notes by Larry Moss and Guram Bezhanishvili 2008

Theorem HALT

The set

HALT = { (M, i) | M is a Turing machine that halts on input i}

is undecidable.

Outline of Turing's proof

Julius Richard Büchi
[1924 -1984]
1961

Turing introduced the Turing machines and proved first:

Reduction from the halting problem to unsatisfiability

Lecture notes by Larry Moss and Guram Bezhanishvili 2008

Theorem HALT

The set

$$\text{HALT} = \{ (M, i) \mid M \text{ is a Turing machine that halts on input } i\}$$

is undecidable.

Theorem HALT-e

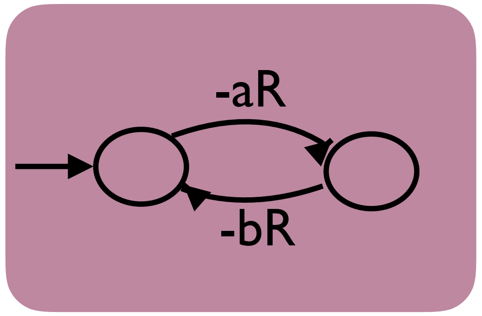
The set

$$\text{HALT-e} = \{ M \mid M \text{ is a Turing machine that halts on empty input}\}$$

is undecidable.

FOL: Reduction

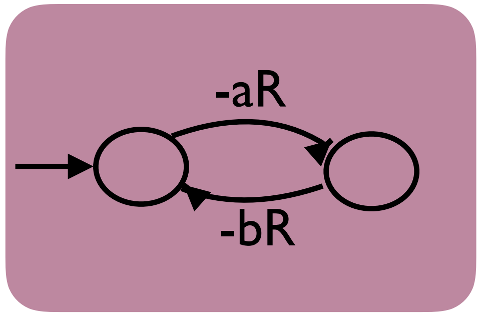
HALT-e
to
UNSAT



$\mapsto$ $\exists x \forall y \ldots \wedge (\forall x \forall z \,..\, \vee \forall x \exists y \,..) \;\wedge \ldots$

FOL: Reduction

HALT-e
to
UNSAT



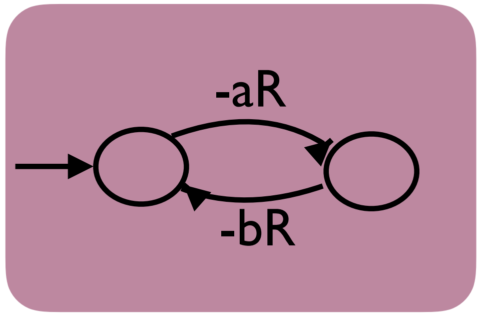
$\mapsto$ $\exists x \forall y \ldots \wedge (\forall x \forall z .. \vee \forall x \exists y ..) \wedge \ldots$

FOL: Reduction

HALT-e to UNSAT

Reduction Theorem

For a Turing machine M, we construct a FOL formula φ_M such that φ_M is unsatisfiable iff M halts on empty input.



$\exists x \forall y \ \ldots \wedge (\forall x \forall z \ldots \vee \forall x \exists y \ldots) \ \wedge \ldots$

FOL: Reduction

HALT-e to UNSAT

Reduction Theorem

For a Turing machine M, we construct a FOL formula φ_M such that φ_M is unsatisfiable iff M halts on empty input.

Corollary - Undecidability of FOL

UNSAT = $\{\varphi \mid \varphi$ is an unsatisfiable first-order formula $\}$ is undecidable.
Hence, VALID is undecidable too.

Turing machine

= finite control (automaton states)

\+ (potentially) infinite tape

\+ head for reading/writing

unlimited access
to infinite memory

Turing machines

A Turing machine is a tuple $M = (Q, \Sigma, \delta, q_0, q_1)$, where

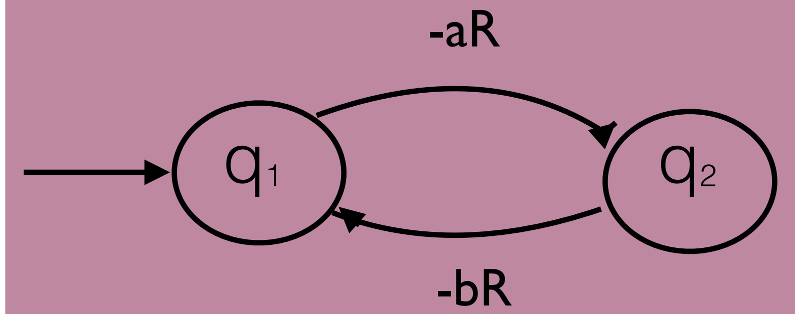
Q is a finite set of states with q_0 the halting state and q_1 the starting state, ($q_0, q_1 \in Q$)

Σ is a finite set, the alphabet, and

$\delta: Q \setminus \{q_0\} \times \Sigma \longrightarrow \Sigma \times \{L, C, R\} \times Q$ is the transition function.

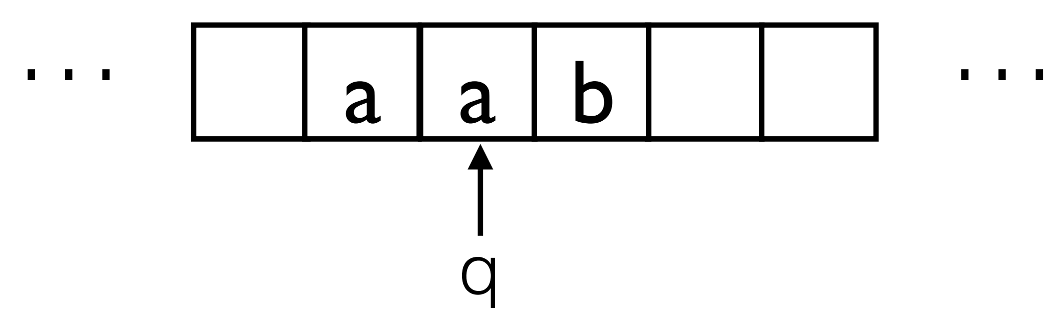
$\delta(q,a) = (b,X,r)$ means that in a state q, reading the symbol a from the cell on which the head is positioned, the TM writes b in place of a, moves the head for one cell in direction X, and changes to state r.

Example



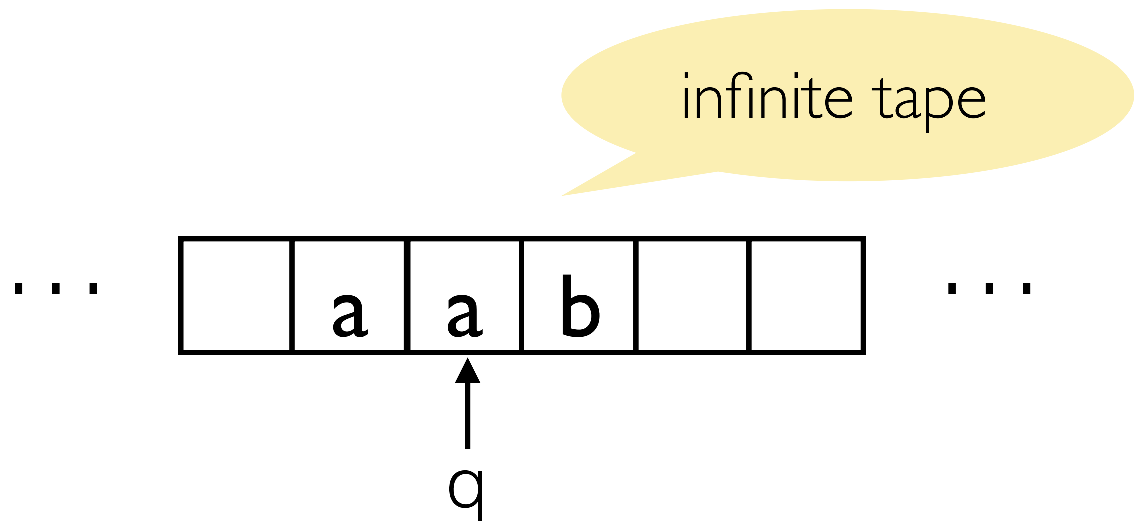
Turing machines

Compute via configurations, triples (q,w,h)
notation uqv where uv = w and |u| = h-1.



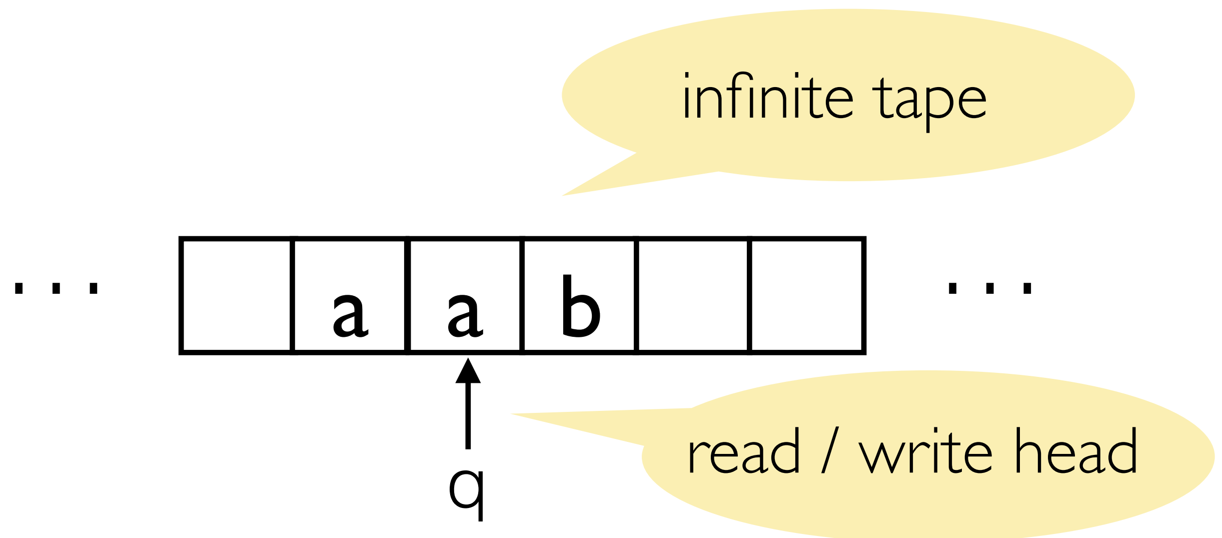
Turing machines

Compute via configurations, triples (q,w,h)
notation uqv where uv = w and |u| = h-1.



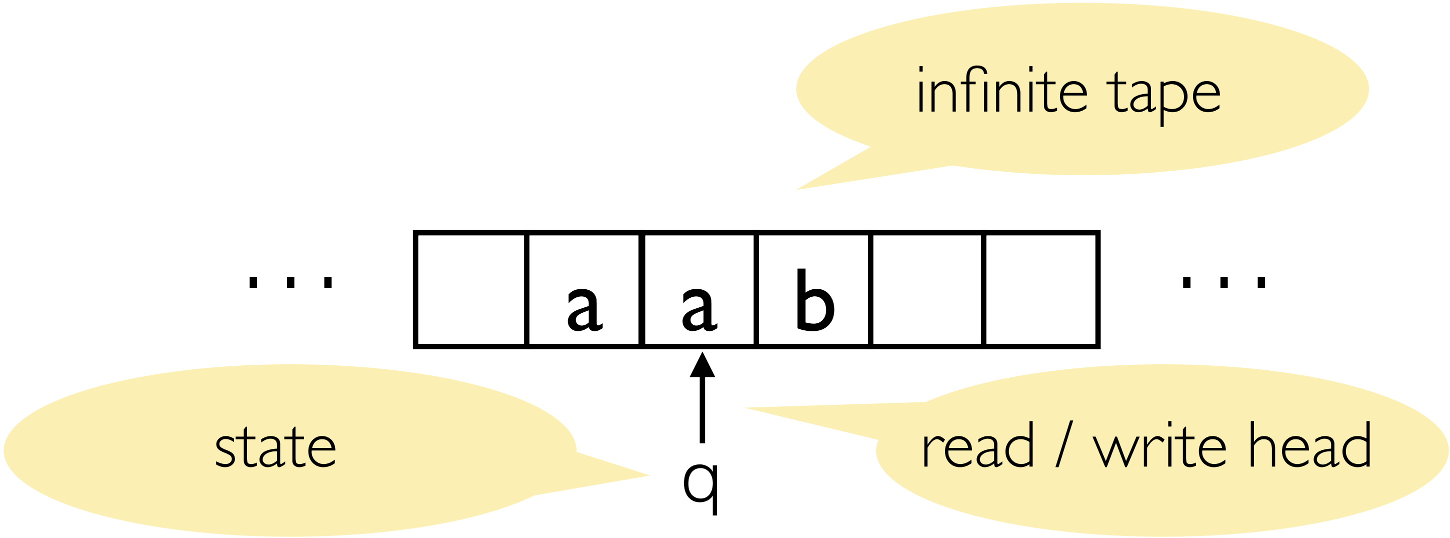
Turing machines

Compute via configurations, triples (q,w,h)
notation uqv where uv = w and |u| = h-1.



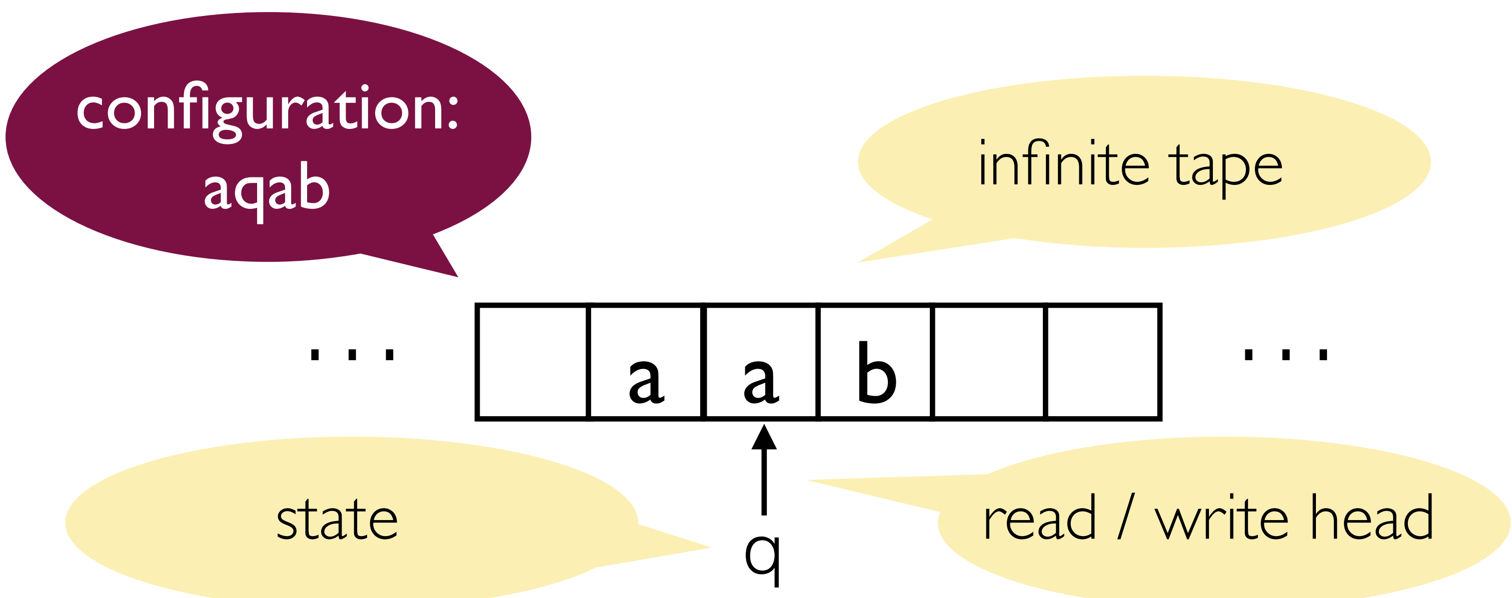
Turing machines

Compute via configurations, triples (q,w,h)
notation uqv where uv = w and |u| = h-1.



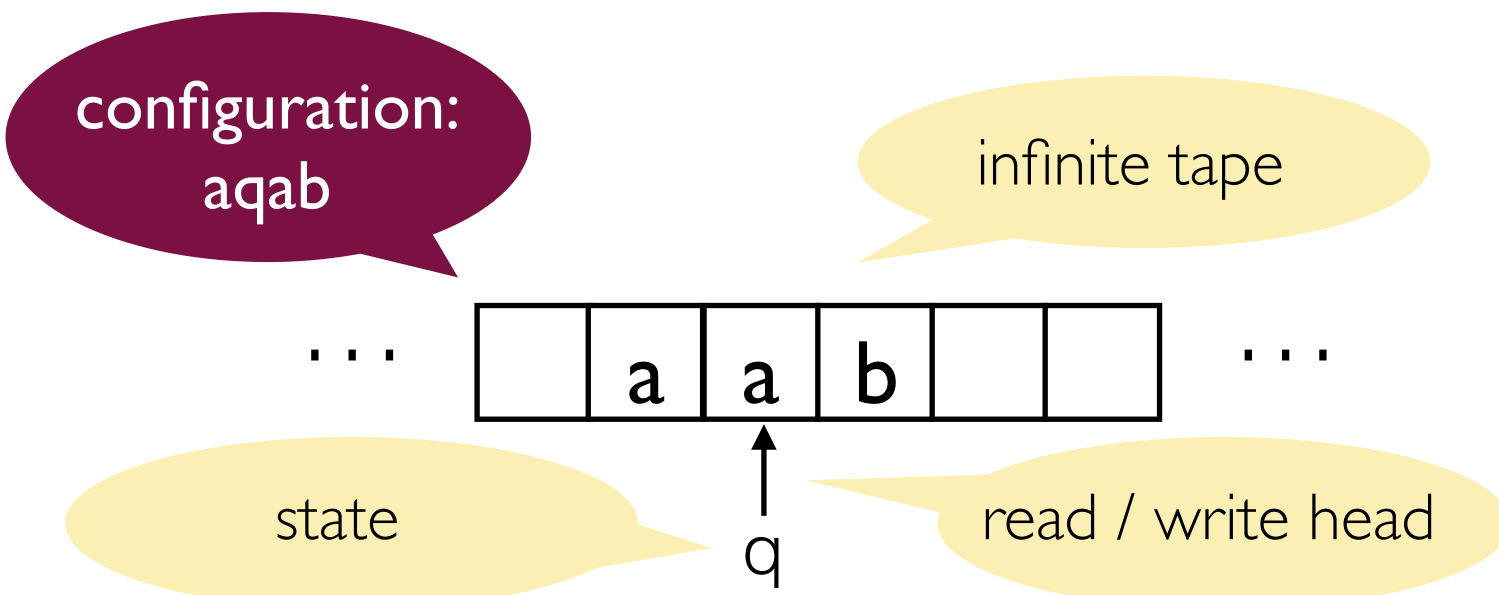
Turing machines

Compute via configurations, triples (q,w,h)
notation uqv where uv = w and |u| = h-1.



Turing machines

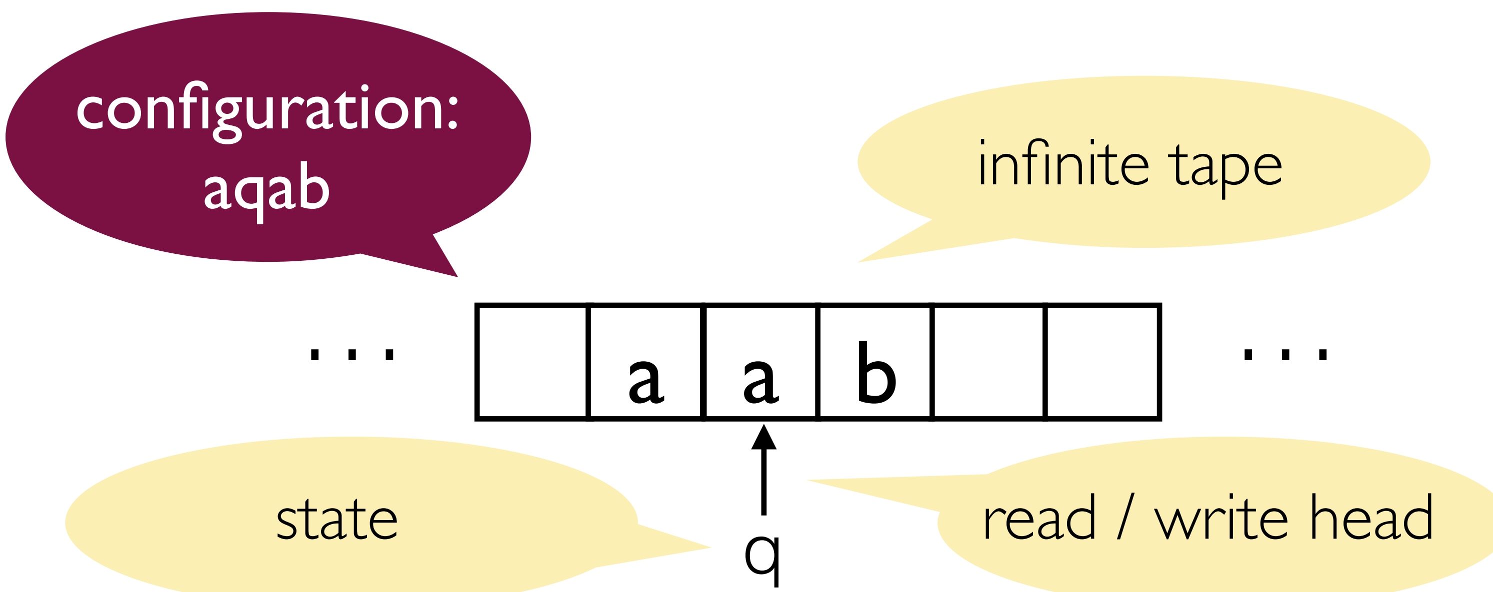
Compute via configurations, triples (q,w,h)
notation uqv where uv = w and |u| = h-1.



$$
\begin{array}{llll}
uaqbv & \models uracv & \text{iff} & \delta(q,b) = (c,L,r) \\
uaqbv & \models uarcv & \text{iff} & \delta(q,b) = (c,C,r) \\
uqbv & \models ucrv & \text{iff} & \delta(q,b) = (c,R,r) \\
qbv & \models r\Box cv & \text{iff} & \delta(q,b) = (c,L,r) \\
qbv & \models rcv & \text{iff} & \delta(q,b) = (c,C,r) \\
uaq & \models uarc & \text{iff} & \delta(q,\Box) = (c,C,r) \\
uaq & \models uacr & \text{iff} & \delta(q,\Box) = (c,R,r)
\end{array}
$$

Turing machines

Compute via configurations, triples (q,w,h)
notation uqv where uv = w and |u| = h-1.



$uaqbv \models uracv$ iff $\delta(q,b) = (c,L,r)$
$uaqbv \models uarcv$ iff $\delta(q,b) = (c,C,r)$
$uqbv \models ucrv$ iff $\delta(q,b) = (c,R,r)$
$qbv \models r\Box cv$ iff $\delta(q,b) = (c,L,r)$
$qbv \models rcv$ iff $\delta(q,b) = (c,C,r)$
$uaq \models uarc$ iff $\delta(q,\Box) = (c,C,r)$
$uaq \models uacr$ iff $\delta(q,\Box) = (c,R,r)$

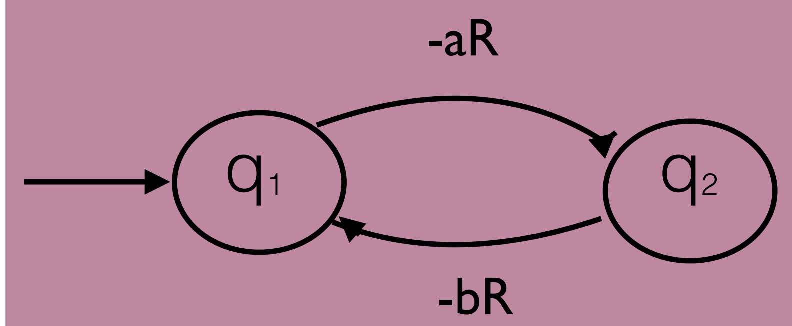
M decides P iff

for all $p \in P$, $q_1p \models^* q_0YES$
for all $p \notin P$, $q_1p \models^* q_0NO$

Example run on empty input

sequence of configurations

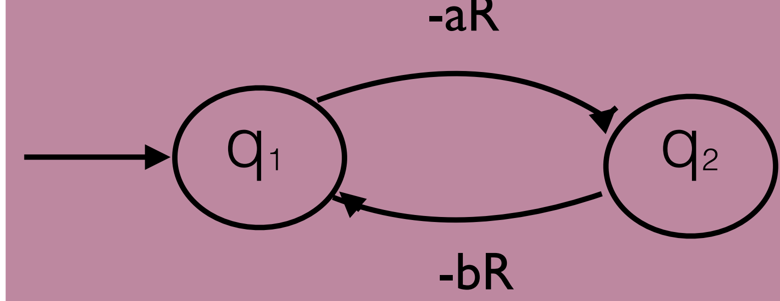
Example

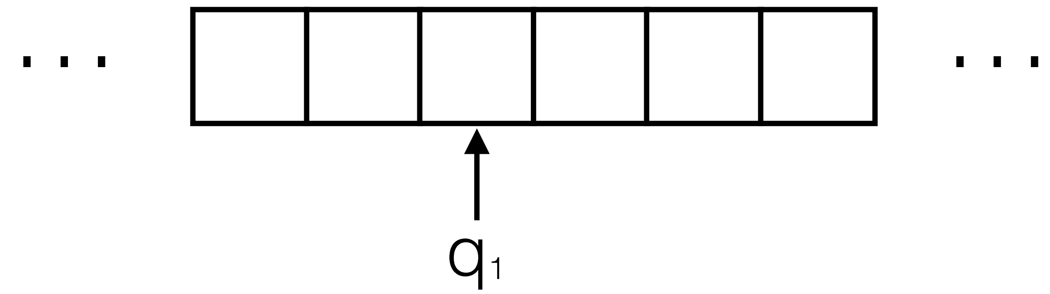


Example run on empty input

sequence of configurations

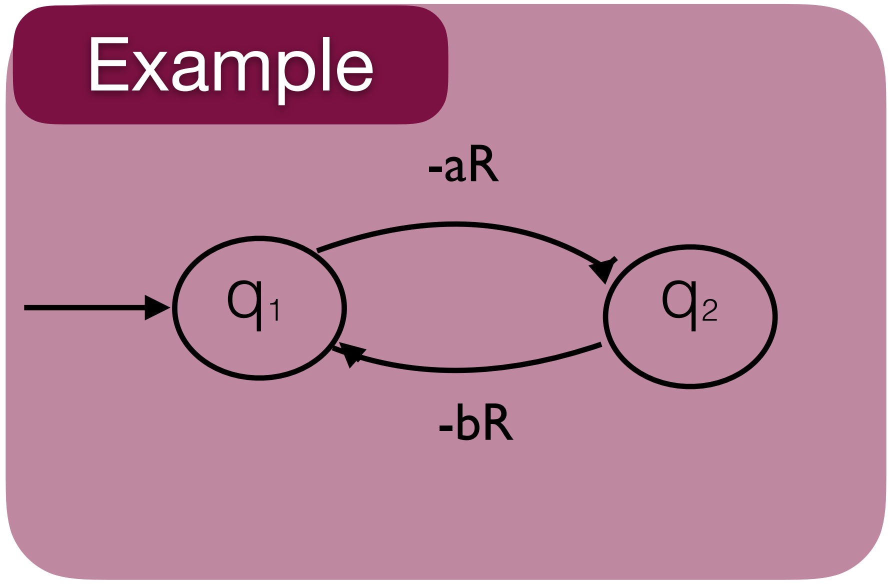
Example

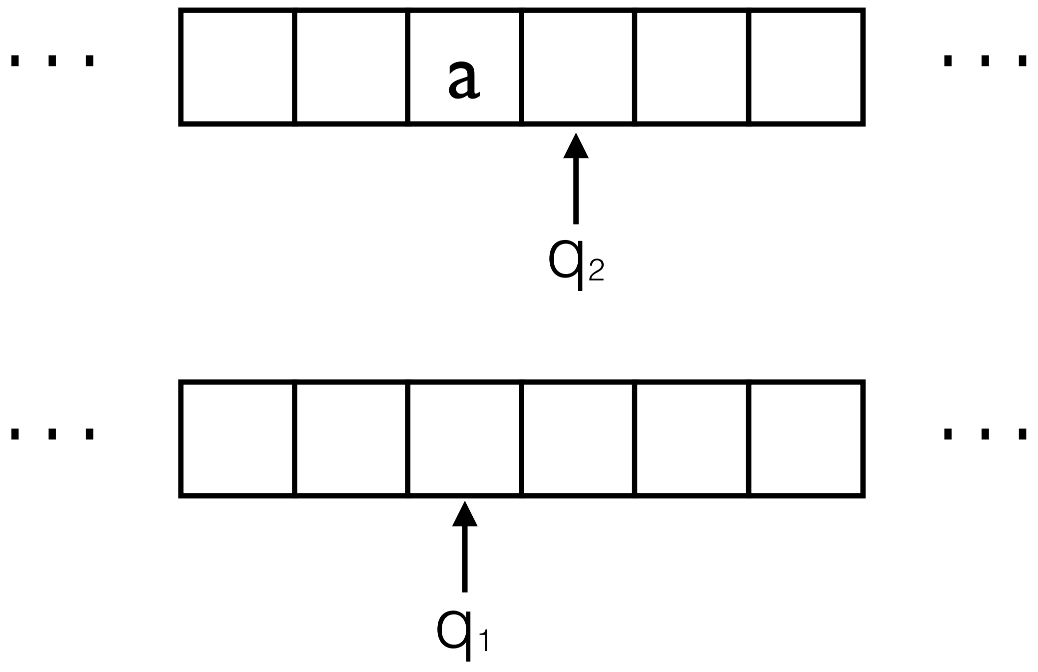




Example run on empty input

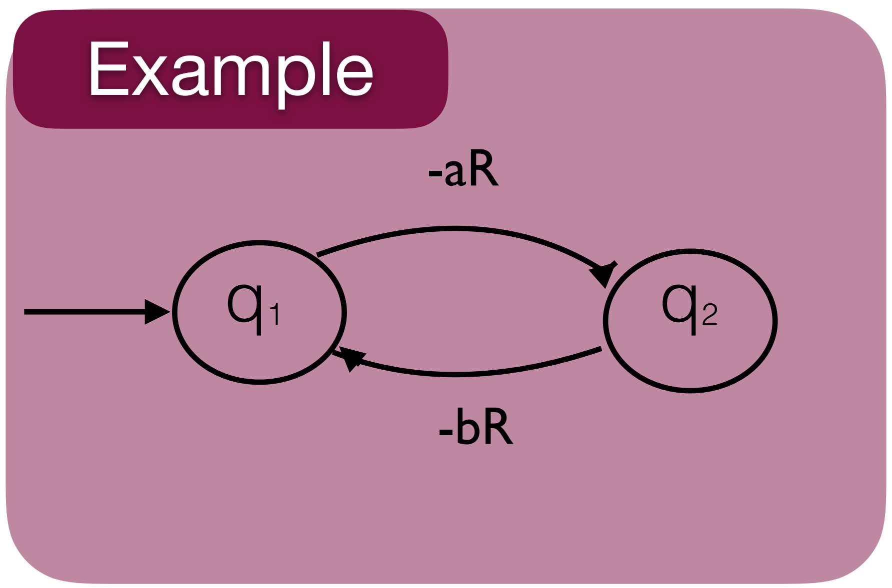
sequence of configurations

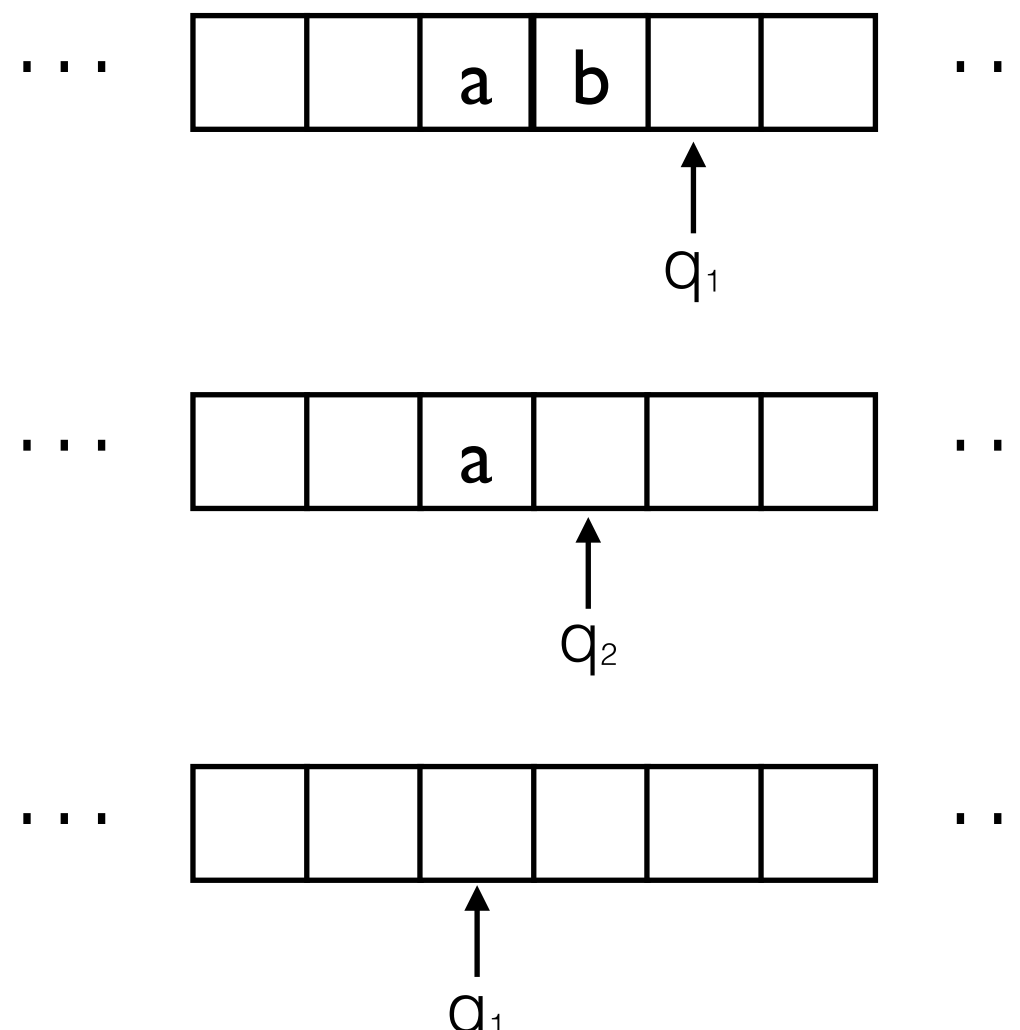




sequence of configurations

Example run on empty input

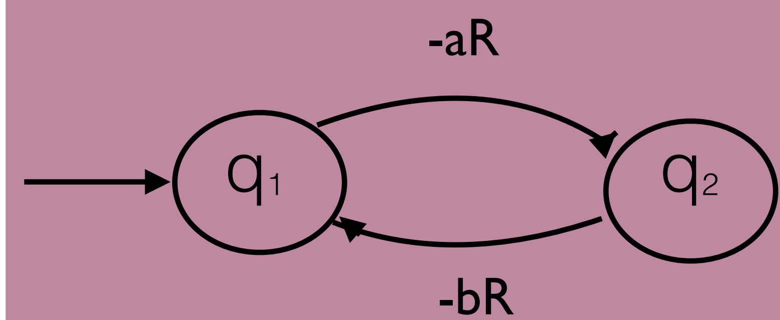


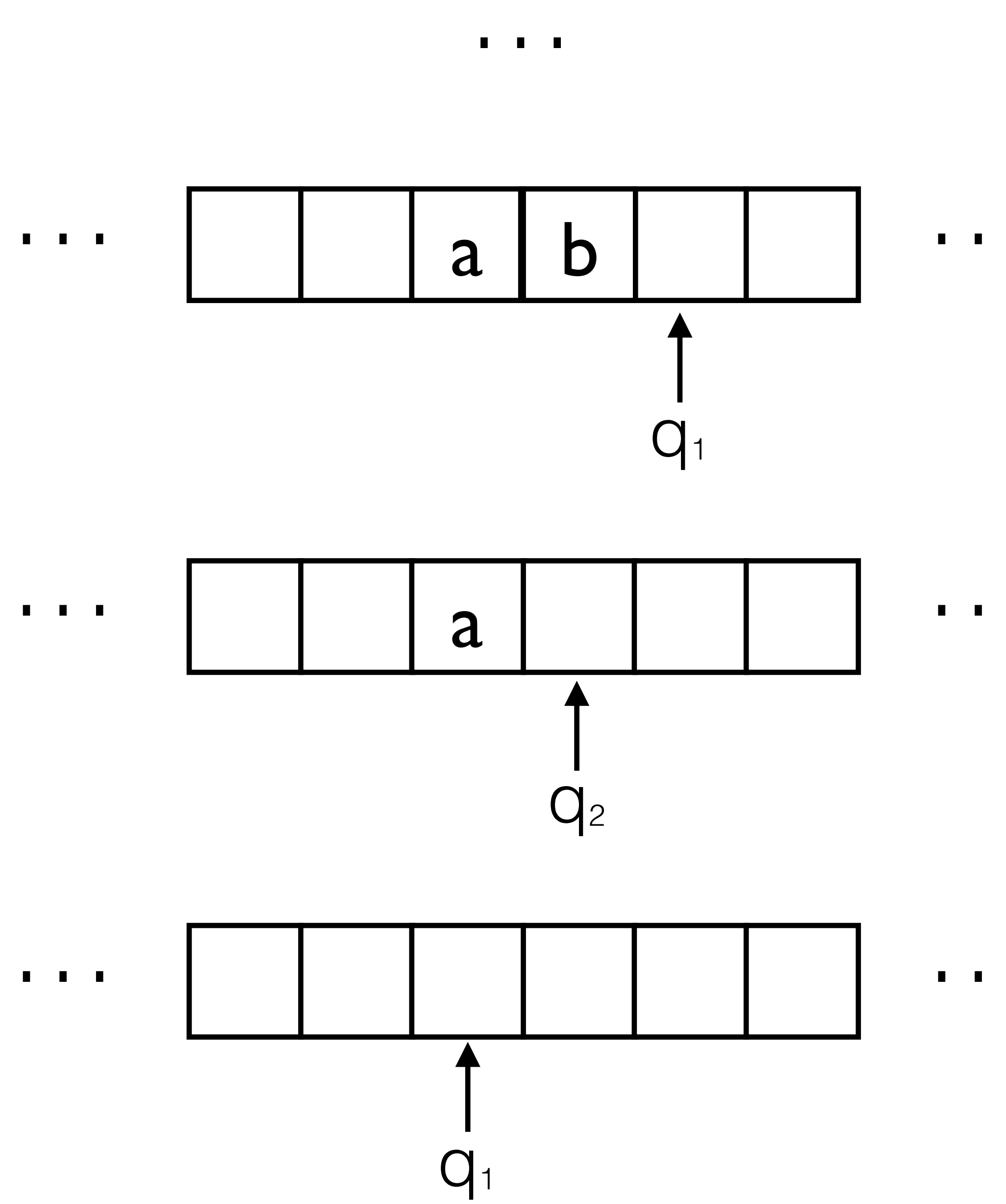


Example run on empty input

sequence of configurations

Example

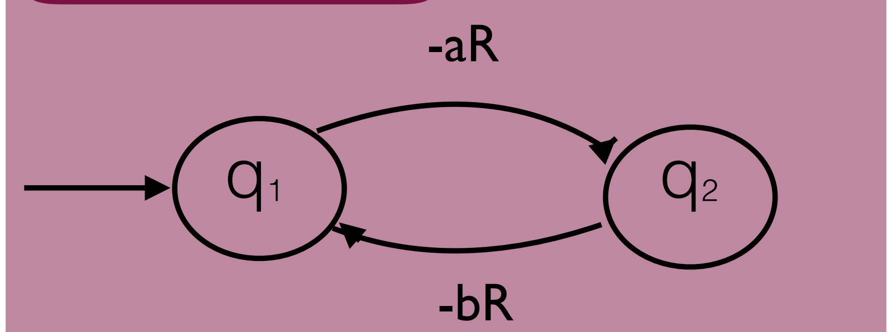


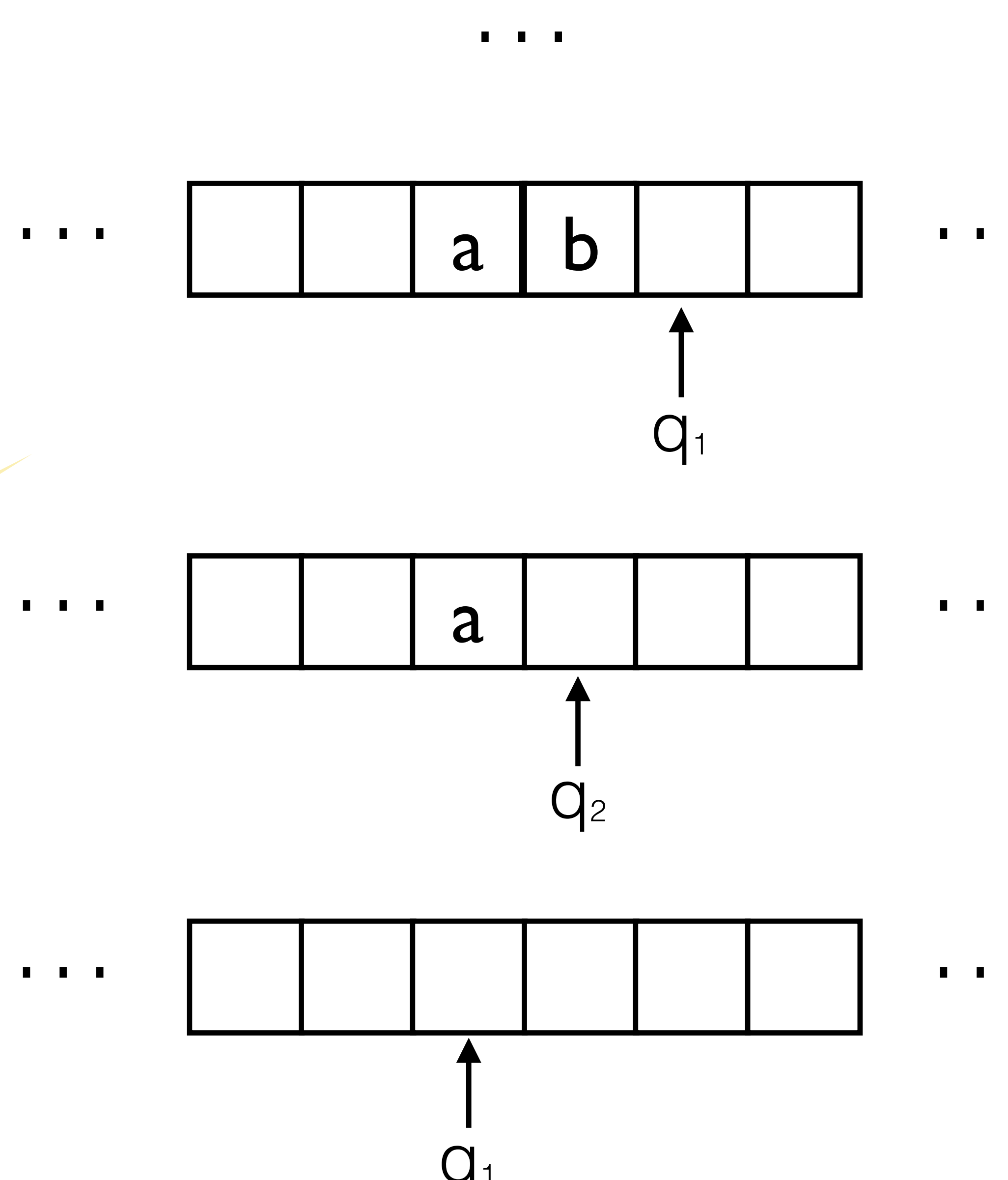


Example run on empty input

sequence of configurations

Example

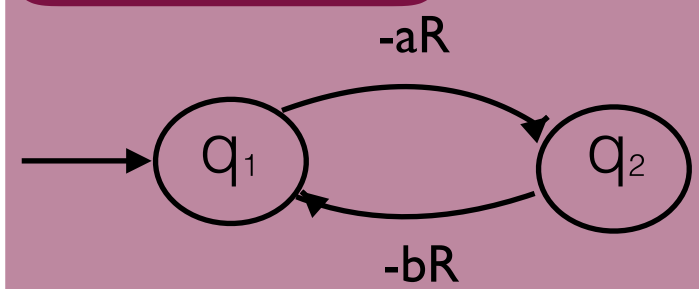




gives us an intended model

The intended model of φ_M

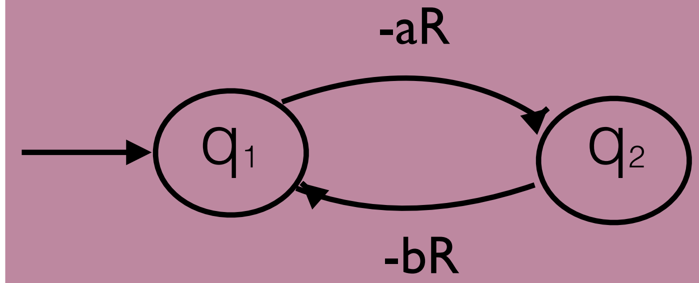
Example



					...						
					S_a	S_b	S_a	S_b	S_a	Q_2	
					S_a	S_b	S_a	S_b	Q_1		
					S_a	S_b	S_a	Q_2			
					S_a	S_b	Q_1				
					S_a	Q_2					
...					Q_1						...

The signature of φ_M

Example



				S_a	S_b	S_a	S_b	S_a	Q_2
				S_a	S_b	S_a	S_b	Q_1	
				S_a	S_b	S_a	Q_2		
				S_a	S_b	Q_1			
				S_a	Q_2				
				Q_1					

Q_i - unary predicate symbol for each $q_i \in Q$

Sa - unary predicate symbol for each $a \in \Sigma$

head - unary predicate symbol

\+ some more function and predicate symbols for describing the upper half plane:

origin - constant symbol

east, west, north - unary function symbols

y-axis, left-side, right-side, bottom - unary predicate symbols

The encoding φ_M

φ_M = upper-half-plane $\wedge$
unique-symbol-per-cell $\wedge$
initial-situation $\wedge$
no-head-no-change $\wedge$
does-not-halt $\wedge$

one-step-formulas $\wedge$

head-movement-formula

The encoding φ$_M$

φ$_M$ = upper-half-plane ∧
unique-symbol-per-cell ∧
initial-situation ∧
no-head-no-change ∧
does-not-halt ∧

one-step-formulas ∧

head-movement-formula

$$\forall x. \bigvee_{a \in \Sigma} (S_a(x) \wedge \bigwedge_{b \neq a} \neg S_b(x))$$

The encoding φM

φM = upper-half-plane ∧
unique-symbol-per-cell ∧
initial-situation ∧
no-head-no-change ∧
does-not-halt ∧

one-step-formulas ∧

head-movement-formula

$$Q_1(\text{origin}) \wedge \bigwedge_{k \neq 1} \neg Q_k(\text{origin}) \wedge$$
$$\forall x.\text{bottom}(x) \rightarrow S_{\square}(x) \wedge (x = \text{origin} \leftrightarrow \text{head}(x))$$

The encoding φ_M

φ_M = upper-half-plane $\wedge$
unique-symbol-per-cell $\wedge$
initial-situation $\wedge$
no-head-no-change $\wedge$
does-not-halt $\wedge$

one-step-formulas $\wedge$

head-movement-formula

Homework

The encoding φ_M

φ_M = upper-half-plane $\wedge$
unique-symbol-per-cell $\wedge$
initial-situation $\wedge$
no-head-no-change $\wedge$
does-not-halt $\wedge$

one-step-formulas $\wedge$

head-movement-formula

$$\forall x. \neg Q_0(x)$$

The encoding φM

φM = upper-half-plane ∧
unique-symbol-per-cell ∧
initial-situation ∧
no-head-no-change ∧
does-not-halt ∧

one-step-formulas ∧

head-movement-formula

$$\forall x.(\text{head}(x) \wedge Q_i(x) \wedge S_a(x)) \rightarrow (S_b(\text{north}(x)) \wedge \text{head}(\text{north} \circ \text{west}(x)) \wedge Q_j(\text{north} \circ \text{west}(x))$$

$$\forall x.(\text{head}(x) \wedge Q_i(x) \wedge S_a(x)) \rightarrow \dots$$

$$\forall x.(\text{head}(x) \wedge Q_i(x) \wedge S_a(x)) \rightarrow \dots$$

The encoding φ_M

φ_M =

upper-half-plane $\wedge$
unique-symbol-per-cell $\wedge$
initial-situation $\wedge$
no-head-no-change $\wedge$
does-not-halt $\wedge$

one-step-formulas $\wedge$

head-movement-formula

$\delta(q_i,a) = (b,L,q_j)$

$$\forall x.(\text{head}(x) \wedge Q_i(x) \wedge S_a(x)) \rightarrow (S_b(\text{north}(x)) \wedge \text{head}(\text{north} \circ \text{west}(x)) \wedge Q_j(\text{north} \circ \text{west}(x))$$

$$\forall x.(\text{head}(x) \wedge Q_i(x) \wedge S_a(x)) \rightarrow \ldots$$

$$\forall x.(\text{head}(x) \wedge Q_i(x) \wedge S_a(x)) \rightarrow \ldots$$

The encoding φM

φM = upper-half-plane ∧
unique-symbol-per-cell ∧
initial-situation ∧
no-head-no-change ∧
does-not-halt ∧

one-step-formulas ∧

head-movement-formula

$$\begin{aligned} &\forall x.\mathrm{head}(x) \wedge Q_i(x) \wedge S_a(x) \rightarrow \\ &x = \mathrm{origin} \vee \\ &\exists y.(\mathrm{north} \circ \mathrm{west}(y) = x \wedge \\ \bigvee_{\delta(q_j,b)=(c,L,q_i)} &(\mathrm{head}(y) \wedge Q_j(y) \wedge S_b(y))) \\ &\vee \ (\mathrm{north}(y) = x \wedge \cdots) \\ &\vee \ (\mathrm{north} \circ \mathrm{east}(y) = x \wedge \cdots)) \end{aligned}$$

The encoding φ_M

φ_M =

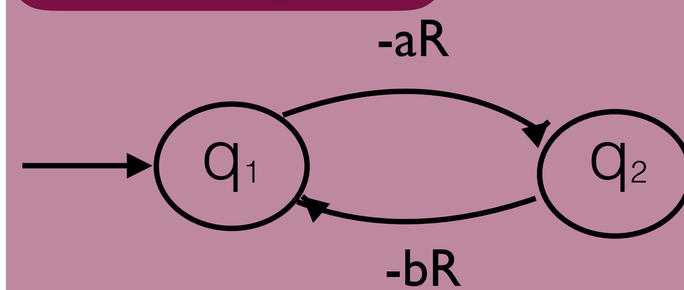
upper-half-plane $\wedge$
unique-symbol-per-cell $\wedge$
initial-situation $\wedge$
no-head-no-change $\wedge$
does-not-halt $\wedge$

one-step-formulas $\wedge$

head-movement-formula

Let's write φ_M for

Example



The encoding φ_M

Next week we will prove

φ_M is satisfiable iff M does not halt on empty input.

$\Leftarrow$ easy:

the intended model is indeed a model

$\Rightarrow$ Simulation Lemma:

if φ_M has a model, then M runs indefinitely

www.cs.uni-salzburg.at/~anas/Undec-FOL.html