Computability

voorjaar 2025

https://liacs.leidenuniv.nl/~vlietrvan1/computability/

college 2, 10 februari 2025

7. Turing Machines

7.1. A General Model of Computation
7.2. Turing Machines as Language Acceptors
7.3. Turing Machines That Compute Partial Functions

A slide from lecture 1

Example 7.7. Accepting $L = \{a^i b a^j \mid 0 \leq i < j\}$

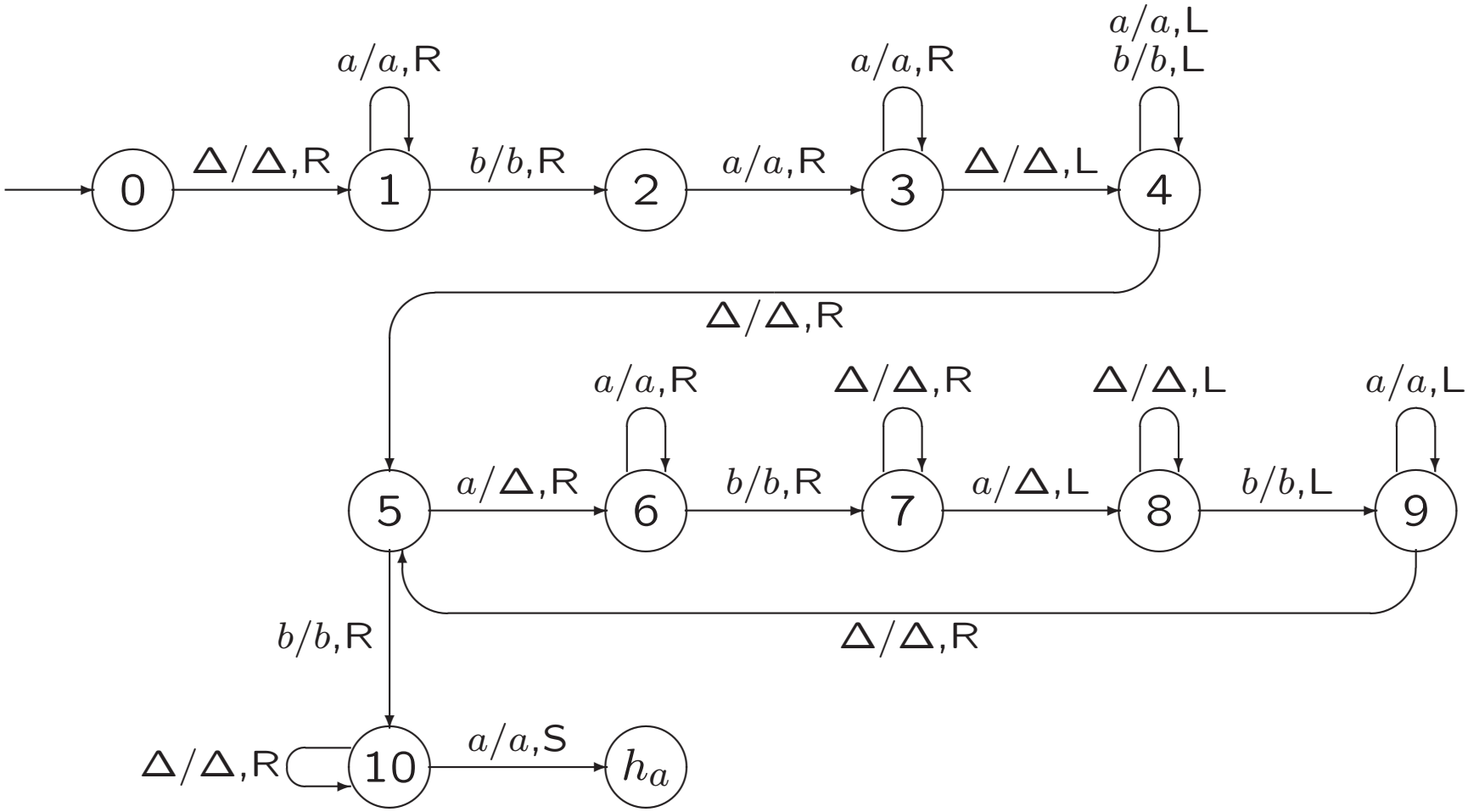
A slide from lecture 1

Example 7.7. Accepting $L = \{a^i b a^j \mid 0 \leq i < j\}$

$$\underline{\triangle}\, a\ \ a\, b\, a\ \ a\ \ a\ \ a$$
$$\triangle\triangle\, a\, b\, a\ \ a\ \ a\ \ a$$
$$\triangle\triangle\, a\, b\triangle\, a\ \ a\ \ a$$
$$\triangle\triangle\triangle b\triangle\, a\ \ a\ \ a$$
$$\triangle\triangle\triangle b\triangle\triangle\, a\ \ a$$

A slide from lecture 1

Example 7.7. Accepting $L = \{a^i b a^j \mid 0 \leq i < j\}$



What if $x \notin L$?

A slide from lecture 1

Example 7.7. Accepting $L = \{a^i b a^j \mid 0 \leq i < j\}$

To illustrate that a Turing machine T may run forever for an input that is not in $L(T)$. No problem!

No problem?

Definition 2.11. A Finite Automaton

A *finite automaton* (FA) is a 5-tuple $(Q, \Sigma, q_0, A, \delta)$, where ...

Definition 5.1. A Pushdown Automaton

A *pushdown automaton* (PDA)
is a 7-tuple $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$, where ...

Definition 7.1. Turing machines

A Turing machine (TM) is ...

Part of a slide from lecture 1

Turing machine

Turing machine has a finite alphabet of symbols.
    (actually two alphabets. . .)
Turing machine has a finite number of states.

7.1. A General Model of Computation

Definition 7.1. Turing machines

A Turing machine (TM) is a 5-tuple $T = (Q, \Sigma, \Gamma, q_0, \delta)$, where

Q is a finite set of states. The two *halt* states h_a and h_r are not elements of Q.

Σ, the input alphabet, and Γ, the tape alphabet, are both finite sets, with $\Sigma \subseteq \Gamma$. The *blank* symbol Δ is not an element of Γ.

q_0, the initial state, is an element of Q.

δ is the transition function: ...

Part of a slide from lecture 1

Assumptions about a human computer working with a pencil and paper:

1. . . .

2. Each step taken by the computer depends only on the symbol he is currently examining and on his "state of mind" at the time;

3. . . .

A slide from lecture 1

A move of a Turing machine consists of:

1. Changing from the current state to another, possibly different state;

2. Replacing the symbol in the current square by another, possibly different symbol;

3. Leaving the tape head on the current square, or moving it one square to the right, or moving it one square to the left if it is not already on the leftmost square.

$$\delta(p, X) = (q, Y, D)$$

$$f(n) = n!$$

$$f(x, y) = x * y$$

Definition 7.1. Turing machines

A Turing machine (TM) is a 5-tuple $T = (Q, \Sigma, \Gamma, q_0, \delta)$, where

Q is a finite set of states. The two *halt* states h_a and h_r are not elements of Q.

Σ, the input alphabet, and Γ, the tape alphabet, are both finite sets, with $\Sigma \subseteq \Gamma$. The *blank* symbol Δ is not an element of Γ.

q_0, the initial state, is an element of Q.

δ is the transition function:

$$\delta : Q \times (\Gamma \cup \{\Delta\}) \to (Q \cup \{h_a, h_r\}) \times (\Gamma \cup \{\Delta\}) \times \{R, L, S\}$$

If q is h_a or h_r, the move causes T to halt

What if $D = L$ and T is on square 0?

Normally, TM starts with

- input string starting in square 1 and all other squares blank,
- and its tape head on square 0.

Tape always contains finite number of nonblanks.

Notation:

configuration. . .

Notation:

description of tape contents: $x\underline{\sigma}y$ or $x\underline{y}$
$x\underline{y} = x\underline{y}\Delta = x\underline{y}\Delta\Delta$
if $y = \Lambda$, then $x\underline{\Delta}$

Notation:

description of tape contents: $x\underline{\sigma y}$ or $x\underline{y}$
$x\underline{y} = x\underline{y}\Delta = x\underline{y}\Delta\Delta$
if $y = \Lambda$, then $x\underline{\Delta}$

configuration $xqy = xqy\Delta = xqy\Delta\Delta$
if $y = \Lambda$, then $xq\Delta$

$(Q \cup \{h_a, h_r\}) \cap (\Gamma \cup \{\Delta\}) = \emptyset$

Notation:

description of tape contents: $x\underline{\sigma}y$ or $x\underline{y}$
$x\underline{y} = x\underline{y}\Delta = x\underline{y}\Delta\Delta$
if $y = \Lambda$, then $x\underline{\Delta}$

configuration $xqy = xqy\Delta = xqy\Delta\Delta$
if $y = \Lambda$, then $xq\Delta$

move: $xqy \vdash_T zrw$ $xqy \vdash_T^* zrw$
$xqy \vdash zrw$ $xqy \vdash^* zrw$

example: configuration $aabqa\Delta a$ and $\delta(q, a) = (r, \Delta, L)$ ...

initial configuration corresponding to input x: ...

Notation:

description of tape contents: $x\underline{\sigma}y$ or $x\underline{y}$
$x\underline{y} = x\underline{y}\Delta = x\underline{y}\Delta\Delta$
if $y = \Lambda$, then $x\underline{\Delta}$

configuration $xqy = xqy\Delta = xqy\Delta\Delta$
if $y = \Lambda$, then $xq\Delta$

move: $xqy \vdash_T zrw$ $xqy \vdash_T^* zrw$
$xqy \vdash zrw$ $xqy \vdash^* zrw$

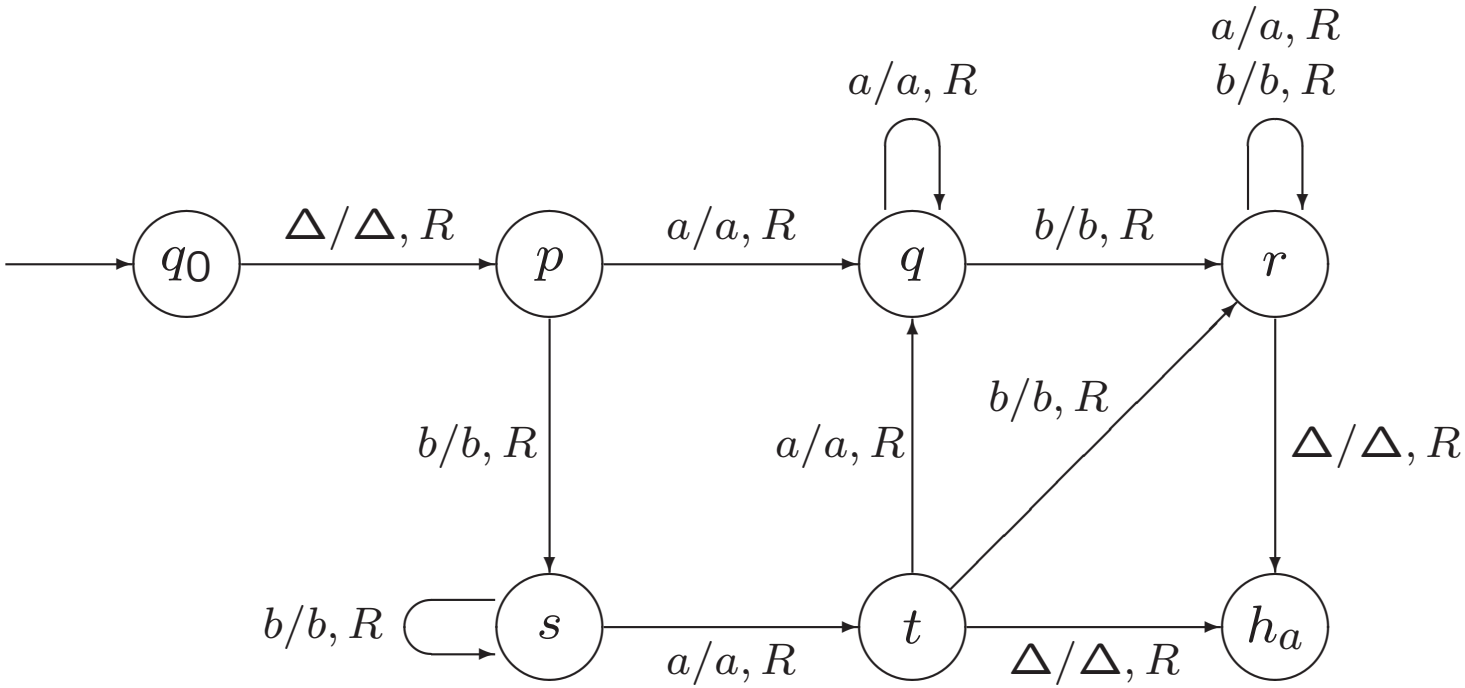
example: configuration $aabqa\Delta a$ and $\delta(q, a) = (r, \Delta, L)$
$aabqa\Delta a \vdash aarb\Delta\Delta a$

initial configuration corresponding to input x: $q_0\Delta x$

A slide from lecture 1

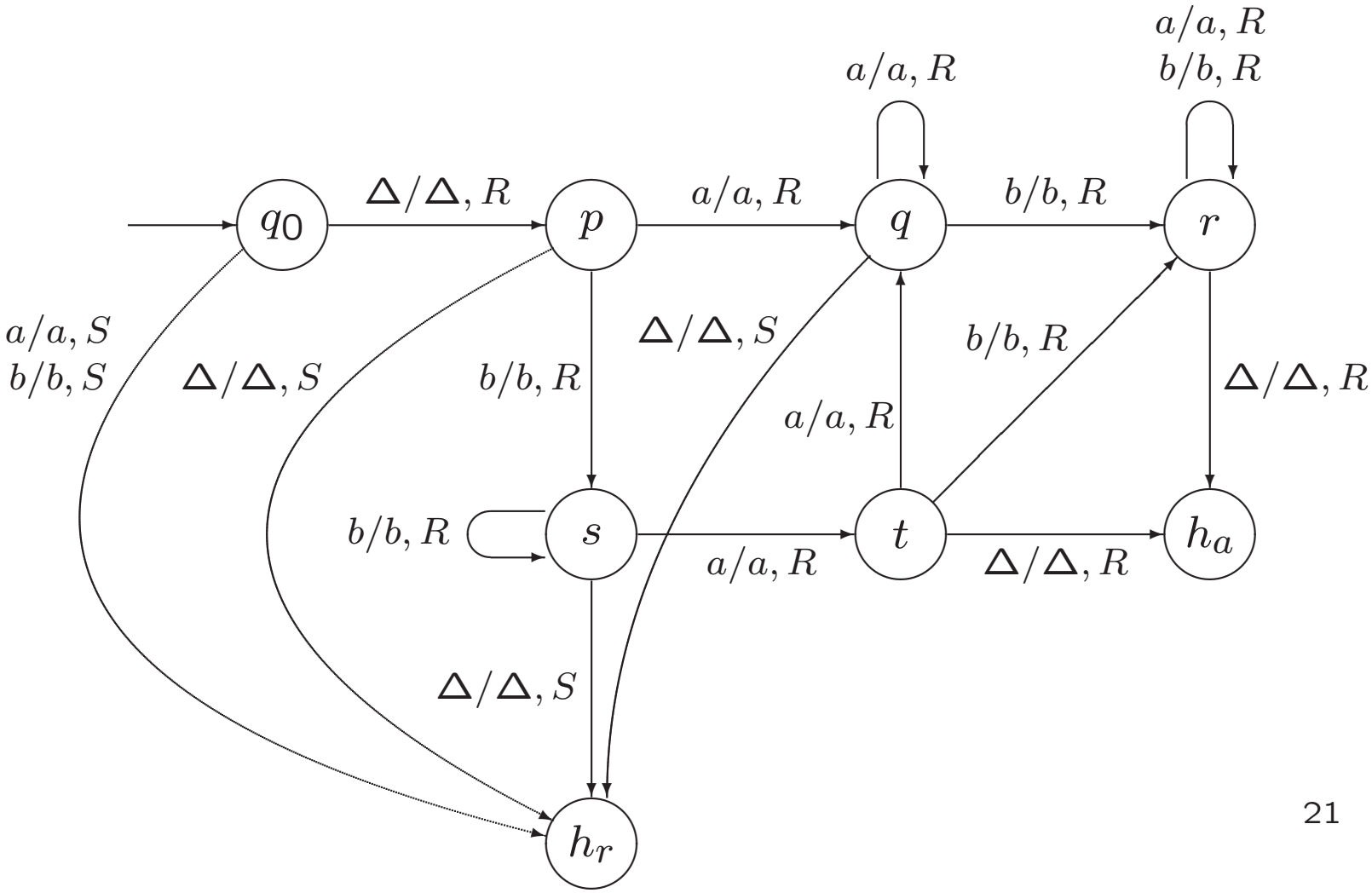
Example 7.3. A TM Accepting a Regular Language

$L = \{a, b\}^*\{ab\}\{a, b\}^* \cup \{a, b\}^*\{ba\}$

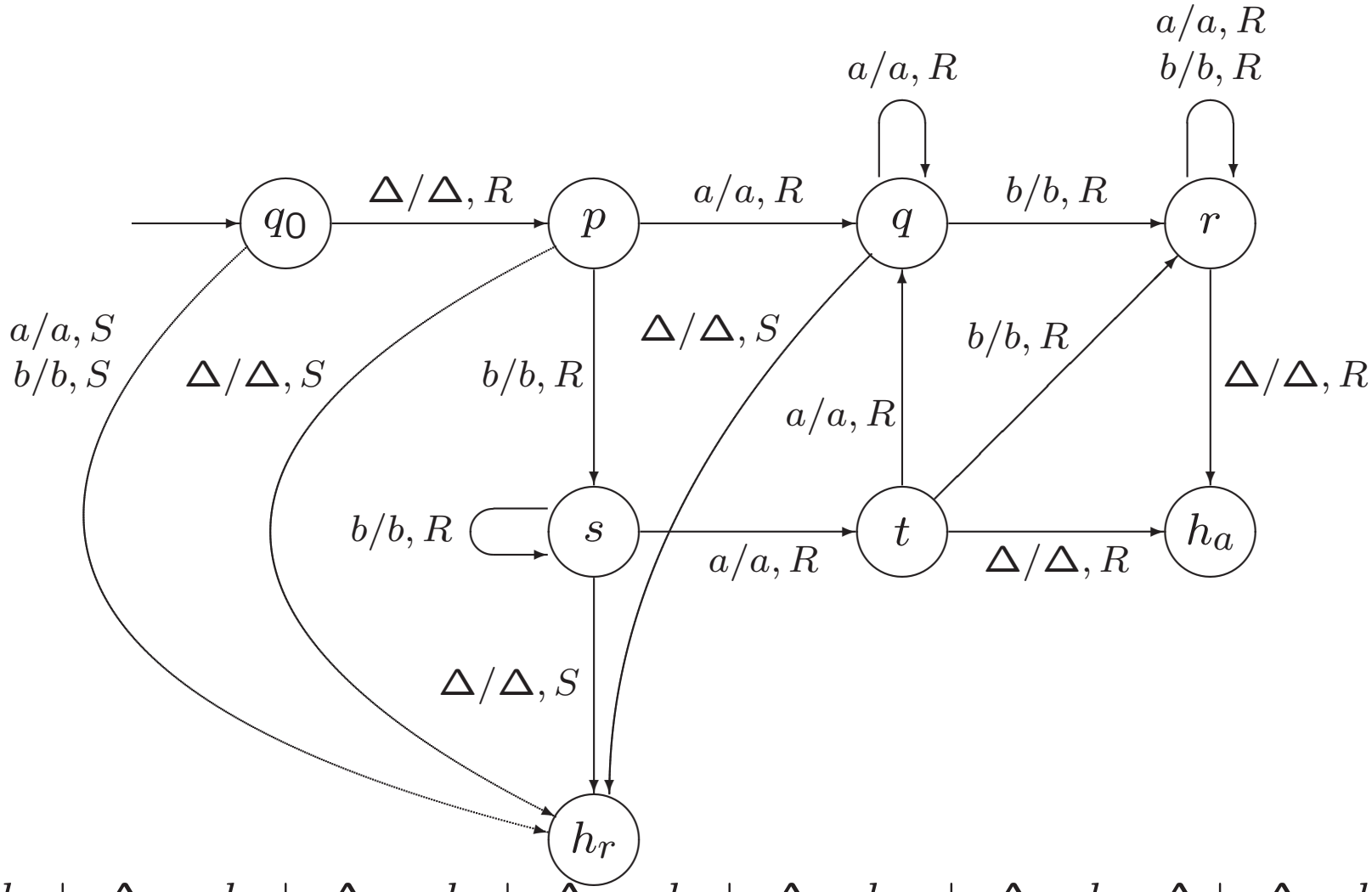


Example 7.3. A TM Accepting a Regular Language

$L = \{a, b\}^*\{ab\}\{a, b\}^* \cup \{a, b\}^*\{ba\}$

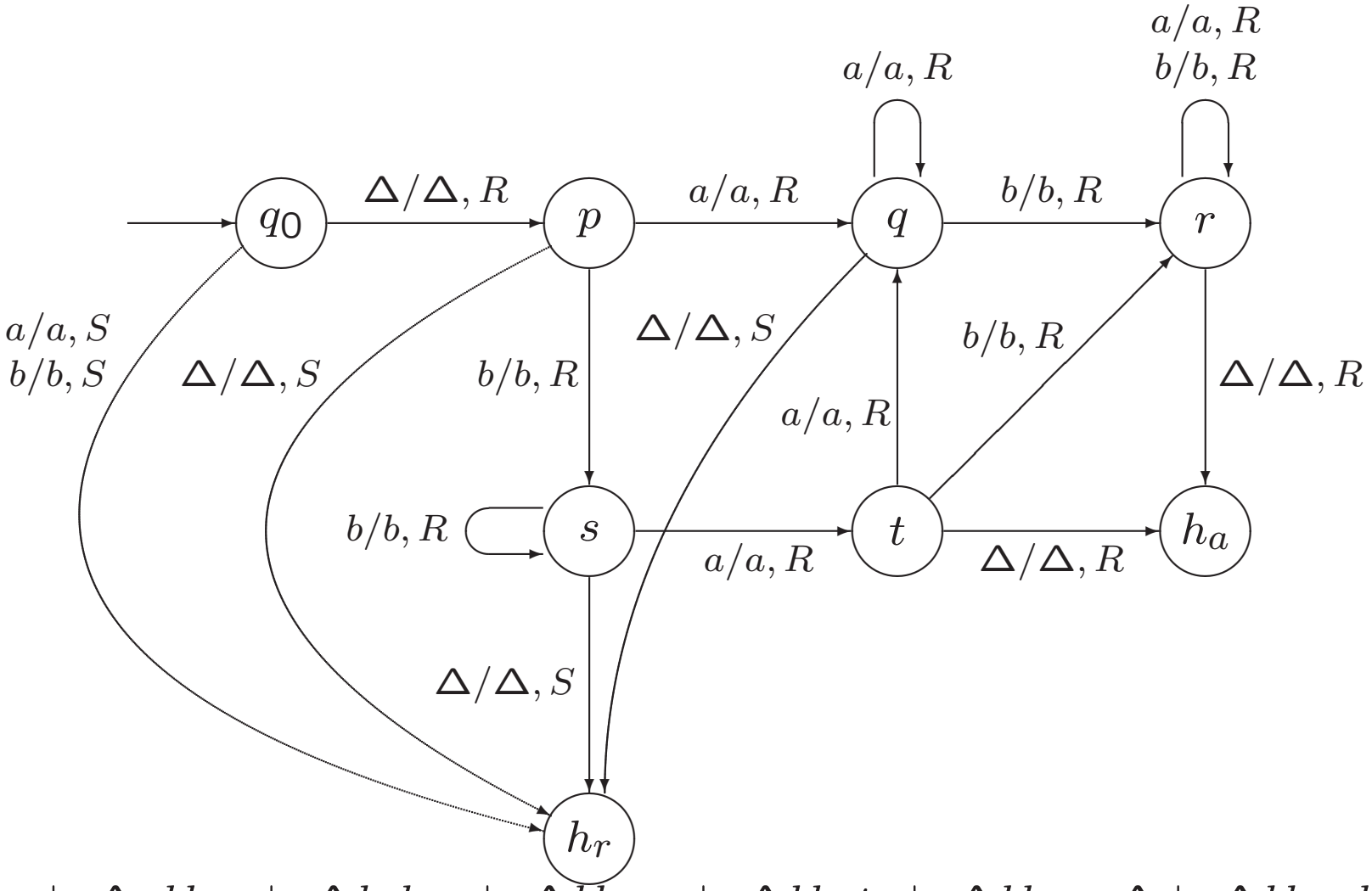


Example 7.3. A TM Accepting a Regular Language



$q_0\Delta aaba \vdash \Delta paaba \vdash \Delta aqaba \vdash \Delta aaqba \vdash \Delta aabra \vdash \Delta aabar\Delta \vdash \Delta aaba\Delta h_a\Delta$

Example 7.3. A TM Accepting a Regular Language



$q_0\Delta bbaa \vdash \Delta pbbaa \vdash \Delta bsbaa \vdash \Delta bbsaa \vdash \Delta bbata \vdash \Delta bbaaq\Delta \vdash \Delta bbaah_r\Delta$

Properties of notation xqy

- all information about configuration in one string
- move TM yields only local change

Useful, when . . .

7.2. Turing Machines as Language Acceptors

Definition 7.2. Acceptance by a TM

If $T = (Q, \Sigma, \Gamma, q_0, \delta)$ is a TM and $x \in \Sigma^*$,
x is accepted by T if

$$q_0 \Delta x \vdash_T^* w h_a y$$

for some strings $w, y \in (\Gamma \cup \{\Delta\})^*$
(i.e., if, starting in the initial configuration corresponding to input x, T eventually halts in the accepting state).

N.B.: sequence of moves leading to h_a is unique

A language $L \subseteq \Sigma^*$ is accepted by T if $L = L(T)$, where

$$L(T) = \{x \in \Sigma^* \mid \ x \text{ is accepted by } T \ \}$$

If $x \notin L(T)$, ...

7.3. Turing Machines That Compute Partial Functions

$$f(x, y) = x/y$$

with real numbers

$f(x, y)$ is not defined if $y = 0$,
i.e., if $y = 0$, then (x, y) is not in domain of f

7.3. Turing Machines That Compute Partial Functions

Example 7.10. The Reverse of a String

$$\underline{\triangle}aabab$$

7.3. Turing Machines That Compute Partial Functions

Example 7.10. The Reverse of a String

$$
\begin{array}{l}
\underline{\triangle}\, a\ a\ b\ a\ b \\
\triangle A\, a\ b\ a\ b \\
\triangle A\, a\ b\ a\, A \\
\triangle B\, a\ b\ a\, A \\
\triangle B A\, b\ a\, A \\
\triangle B A\, b\, A A \\
\triangle B A\, b\, A A \\
\triangle B A B A A \\
\underline{\triangle}\, b\ a\ b\ a\ a
\end{array}
$$

Definition 7.1. Turing machines

A Turing machine (TM) is a 5-tuple $T = (Q, \Sigma, \Gamma, q_0, \delta)$, where

Q is a finite set of states. The two *halt* states h_a and h_r are not elements of Q.

Σ, the input alphabet, and Γ, the tape alphabet, are both finite sets, with $\Sigma \subseteq \Gamma$. The *blank* symbol Δ is not an element of Γ.

q_0, the initial state, is an element of Q.

δ is the transition function:

$$\delta : Q \times (\Gamma \cup \{\Delta\}) \to (Q \cup \{h_a, h_r\}) \times (\Gamma \cup \{\Delta\}) \times \{R, L, S\}$$

Simple version of:

Definition 7.9. A Turing Machine Computing a Function

Let $T = (Q, \Sigma, \Gamma, q_0, \delta)$ be a Turing machine, and f a partial function on Σ^* with values in Γ^*. We say that T computes f if for every x in the domain of f,

$$q_0 \Delta x \vdash_T^* h_a \Delta f(x)$$

and no other input string is accepted by T.

Definition 7.9. A Turing Machine Computing a Function

Let $T = (Q, \Sigma, \Gamma, q_0, \delta)$ be a Turing machine, k a natural number, and f a partial function on $(\Sigma^*)^k$ with values in Γ^*. We say that T computes f if for every $(x_1, x_2, \ldots, x_k)$ in the domain of f,

$$q_0 \Delta x_1 \Delta x_2 \Delta \ldots \Delta x_k \vdash_T^* h_a \Delta f(x_1, x_2, \ldots, x_k)$$

and no other input that is a k-tuple of strings is accepted by T.

A partial function $f : (\Sigma^*)^k \to \Gamma^*$ is Turing-computable, or simply computable, if there is a TM that computes f.

k can be 0. . .

Functions on natural numbers. . .

Example 7.12. The Quotient and Remainder Mod 2