Universidad Carlos III de Madrid
www.uc3m.es

AUTOMATA THEORY AND FORMAL LANGUAGES

UNIT 7: TURING MACHINE

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

OUTLINE

- Definition of Turing Machine
- Variations of Turing Machines
- Universal Turing Machine
- Additional issues

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

OUTLINE

- **Definition of Turing Machine**
- Variations of Turing Machines
- Universal Turing Machine
- Additional issues

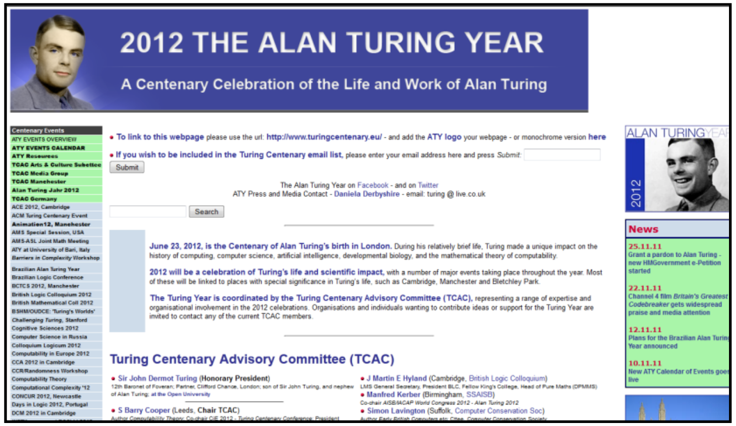
Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Definition of Turing Machine

Alan Turing

http://www.alanturing.net

- 23 June 1912 - 7 June 1954
- Contributions:
  - Mathematics,
  - Cryptanalysis,
  - Logic,
  - …
  - Artificial Intelligence.



http://www.mathcomp.leeds.ac.uk/turing2012/

Este obra está bajo una [licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.]

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Alan Turing

- http://www.alanturing.net
- http://en.wikipedia.org/wiki/Alan_Turing
- Bibliography:
  - "On Computable Numbers, with an Application to the Entscheidungsproblem" 1936
  - Turing, A.M. Computing machinery and intelligence. Mind, 59, 433-460. 1950
    - "I propose to consider the question, "Can machines think?"
    - http://en.wikipedia.org/wiki/Computing_machinery_and_intelligence
    - http://blog.santafe.edu/wp-content/uploads/2009/05/turing1950.pdf
- Charles Petzold. The Annotated Turing: A Guided Tour Through Alan Turing's Historic Paper on Computability and the Turing Machine. Wiley. 2008. www.theannotatedturing.com

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Can machines do everything?

- **We have studied:**
  - Simple languages.
  - Application of simple languages to solve restricted problems:
    - Analysis.
    - Pattern recognition.
    - Syntax analysis.
- **In this unit, we want to answer:**
  - Which languages can be defined by means of any computational device?
  - Which problems can be solved/computed?

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Can machines do everything?

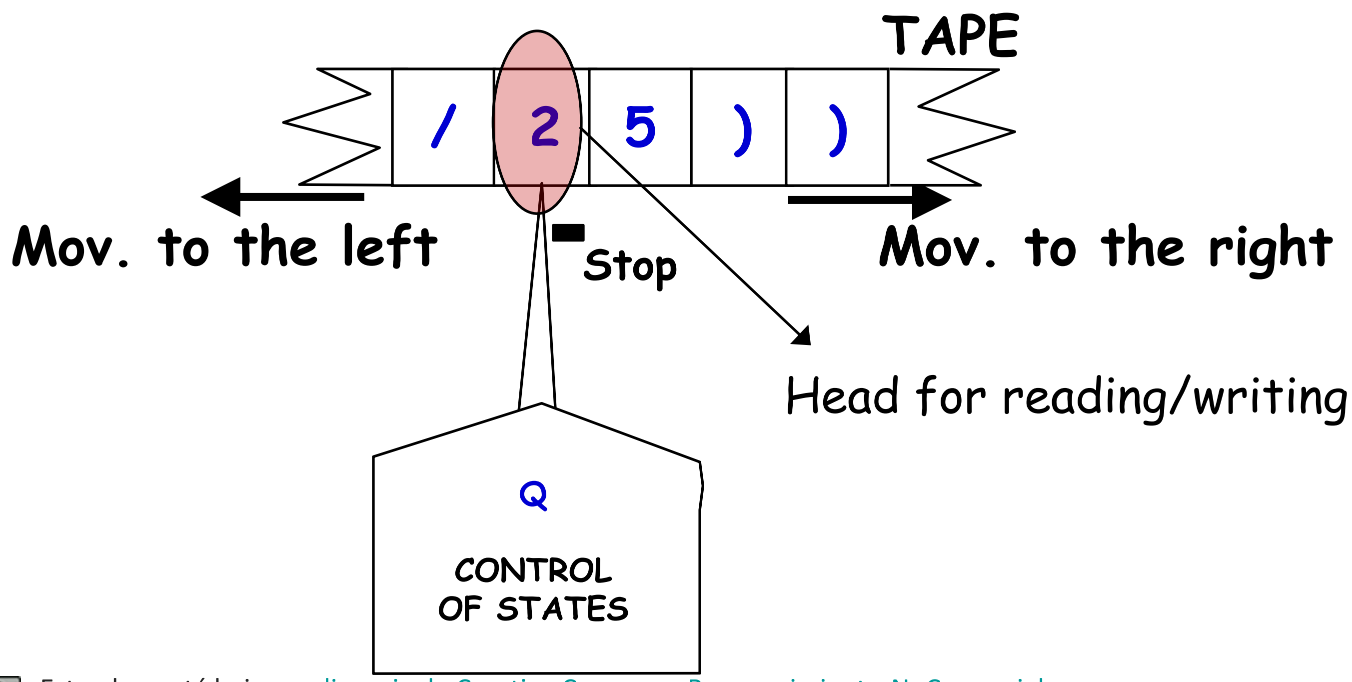
- Which is the solution?
  - He says that she always say the truth.
  - She says that he always lies (he never says the truth).
- Can a computer answer to any possible question in a dialog system?
- Is a grammar ambiguous?
- Is there a solution for $X^N+Y^N=Z^N$, with N>2?

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

What is a Turing Machine?

- Mechanical device:
  - Infinite tape divided into cells with a head for reading/writing.
  - This head can be moved from to left or right or stay in the same cell.



Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Turing Machine: Operation

Operations that a TM carries out:

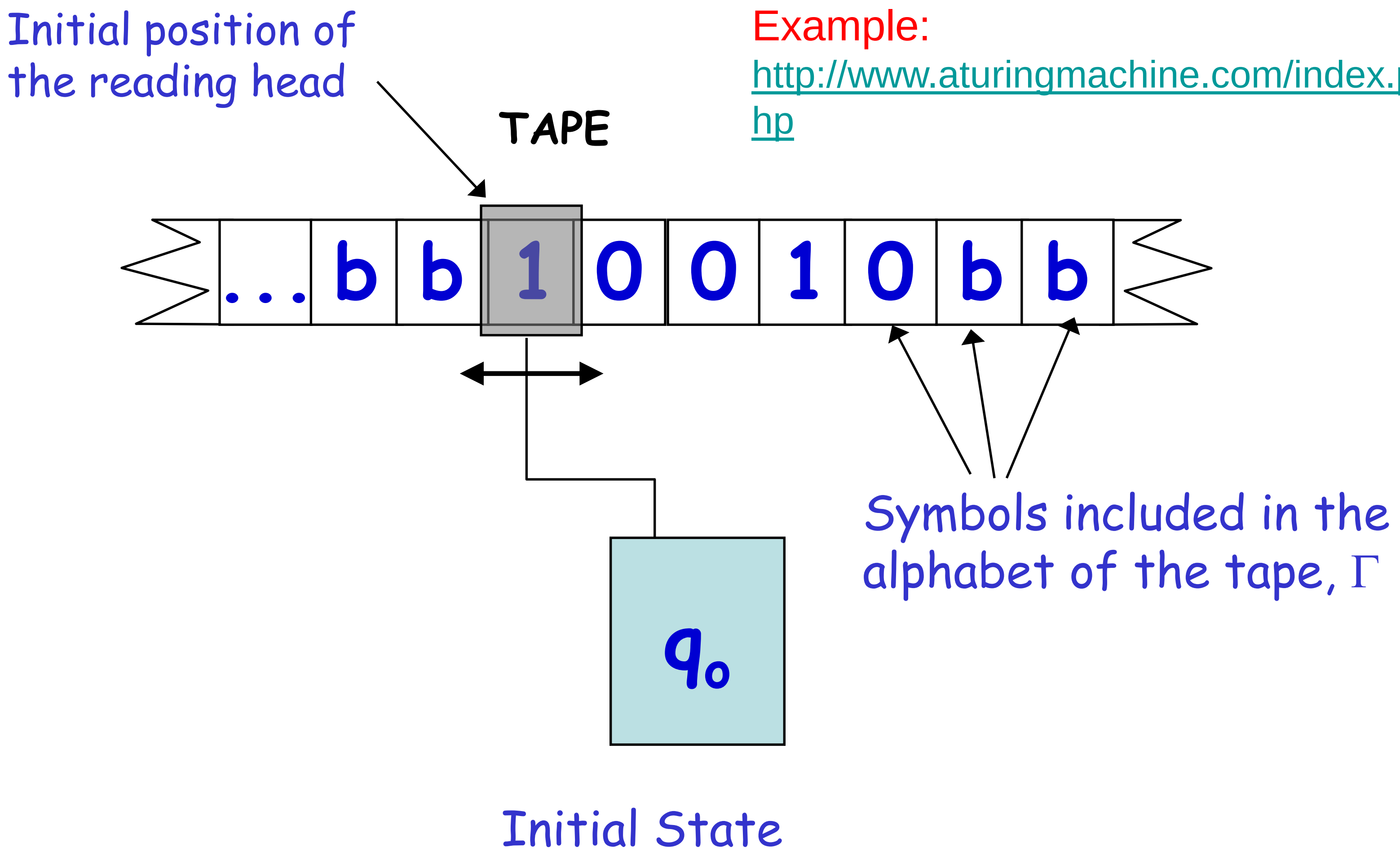
- Being in a state *p* and reading a symbol in the cell on which the R/W head is, it carries out three actions:
  - Transits to a new state.
  - Writes a new symbol in the tape, in the same cell where the current symbol has been read. This symbol replaces the previously read (unless it is the same).
  - Moves the R/W head to the left, to the right or stops in the same position.

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Machine: Initial Situation

Example: http://www.aturingmachine.com/index.php



Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Machine: Operation

Characteristics of the Tape:

- Infinite tape.
- It can contain a character in each cell.
- It can be read.
- It can be written.
- Initially it is considered with infinite blank symbols to the right and left of the word.
- It can be moved to the left or right (a cell each time) or not move.

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Machine: Operation

- There are many variants of Turing Machines…
- … but they all are equivalent.
- The tape is one-dimensional and **infinite** by both sides.
- Initially:
  - The tape contains the word, and the rest of elements of the tape (left and right of the word) are the blank symbol (b or □).
  - At the beginning, the R/W head is located on the left-most element of the word.

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Turing Machine: Formal Definition

- Septuple: **(Σ, Γ, b, Q, qo, f, F),** where:
  - **Γ**: alphabet of symbols in the tape
  - **Σ ⊂ Γ**: alphabet of input symbols
  - **b ∈ Γ, b ∉ Σ** : is the blank symbol (the only symbol allowed to be in the tape infinitely at any step during the computation). It indicates empty cell (□).
  - **Q**: set of states (finite).
  - **q_0 ∈ Q**: initial state.
  - **F ⊂ Q**: set of final states.
  - **f**: transition function

Q x Γ → Q x Γ x {L,R,S}

L or -: left
R or +: right
S or =: stay

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Machine: Transition Function

f: Transition function, table with double input

$$Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R,S\}$$

L: mov. to the left
R: mov. to the right
S: stop

↓Q / Γ→	Symbol	Symbol
Input	State, symbol, movement	…
Input	State, symbol, movement	
,,,	…	

Empty cells in the table:

- Transitions that are NOT possible → The machine stops.

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Turing Machine: Example

$\mathcal{M}$ = (∑={0,1}, Γ ={0,1,□}, □, Q = {q0, q1, qF}, q0, F={qF}, f)

f	0	1	□
-> q0 q1 * qF	(q0, 0, +) (q1, 0, +)	(q1, 1, +) (q0, 1, +)	(qF, 0, =) (qF, 1, =)

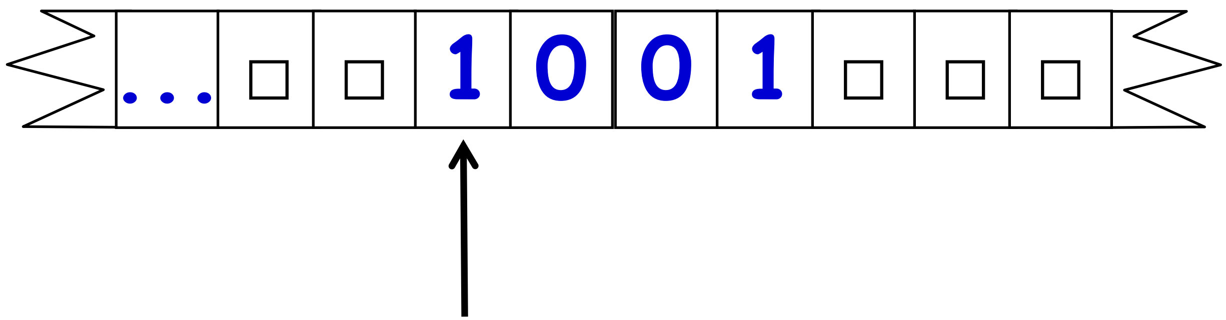
Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Variations of Turing Machines

Turing Machine: Example

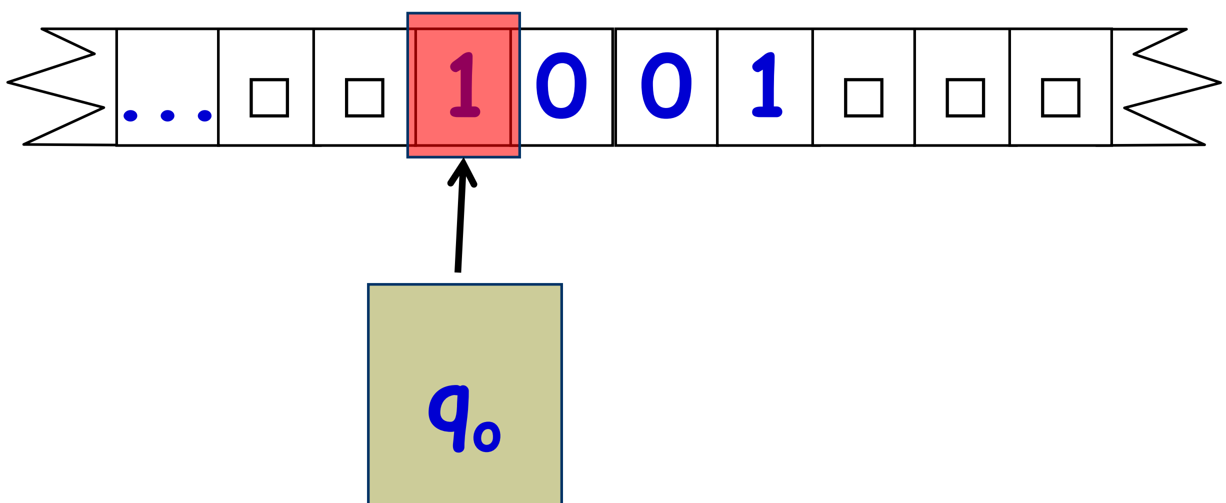
f	0	1	□
-> q0 q1 * qF	(q0, 0, +) (q1, 0, +)	(q1, 1, +) (q0, 1, +)	(qF, 0, =) (qF, 1, =)



Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Turing Machine: Example

f	0	1	□
-> q0	(q0, 0, +)	(q1, 1, +)	(qF, 0, =)
q1	(q1, 0, +)	(q0, 1, +)	(qF, 1, =)
* qF			

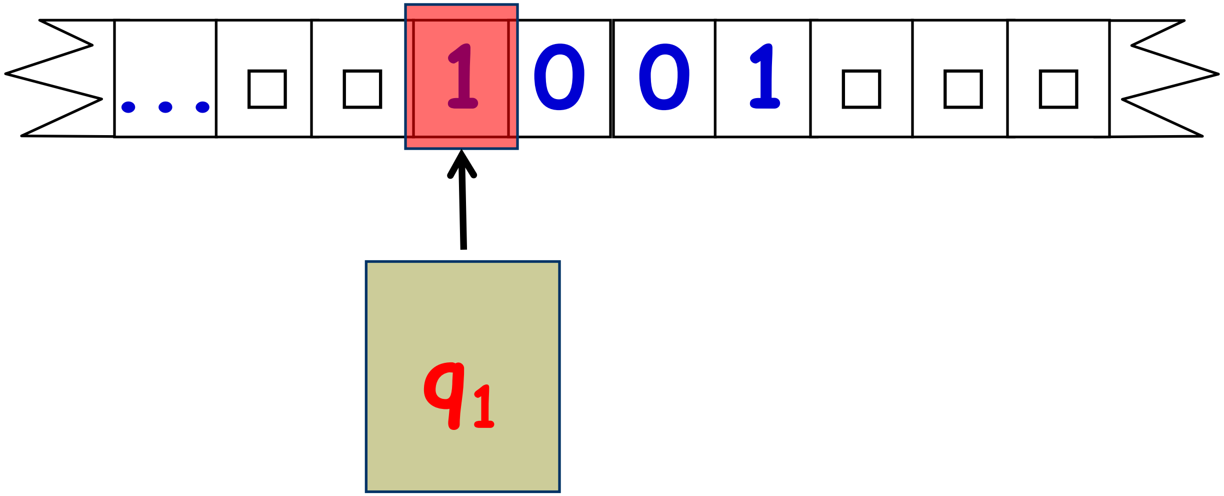


Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Machine: Example

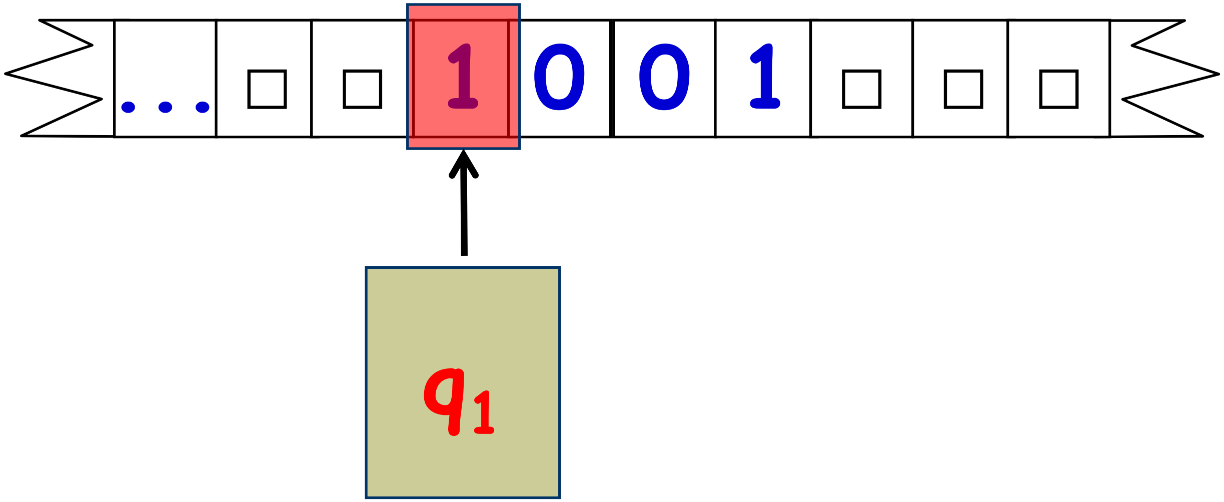
f	0	1	□
-> q0	(q0, 0, +)	(q1, 1, +)	(qF, 0, =)
q1	(q1, 0, +)	(q0, 1, +)	(qF, 1, =)
* qF			



Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Turing Machine: Example

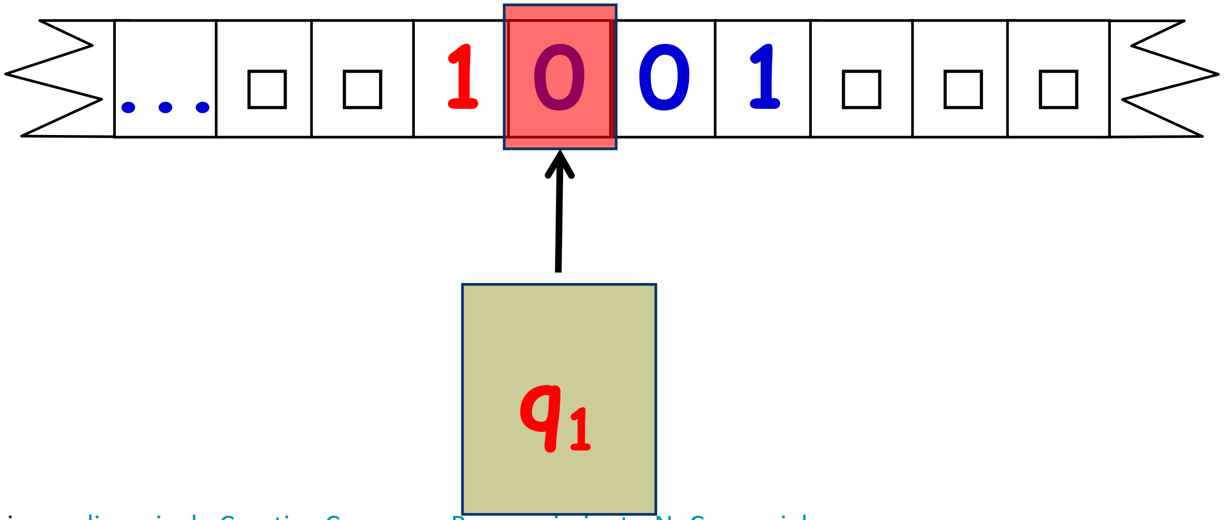
f	0	1	□
-> q0	(q0, 0, +)	(q1, 1, +)	(qF, 0, =)
q1	(q1, 0, +)	(q0, 1, +)	(qF, 1, =)
* qF			



Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Turing Machine: Example

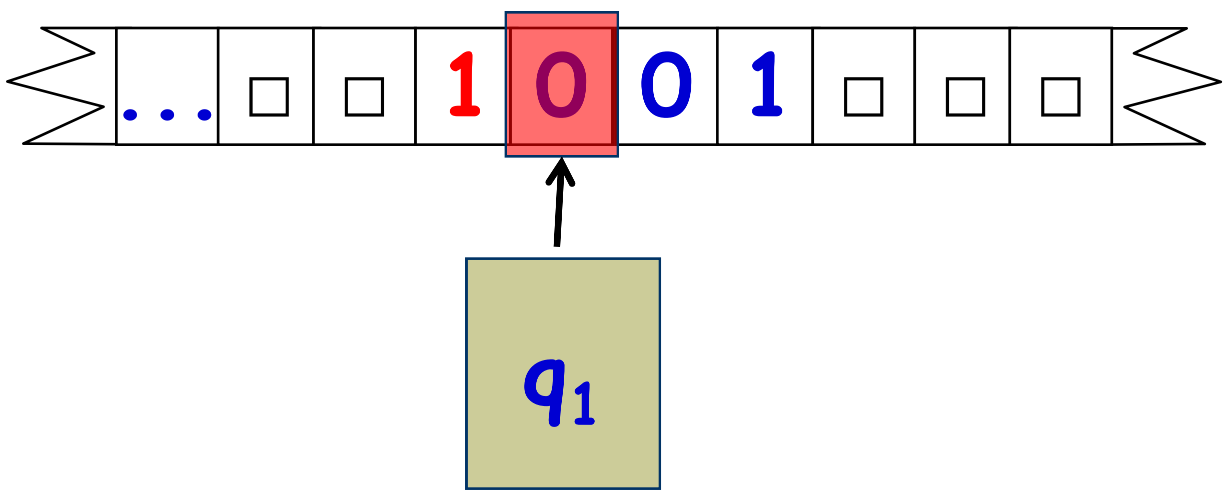
f	0	1	□
-> q0	(q0, 0, +)	(q1, 1, +)	(qF, 0, =)
q1	(q1, 0, +)	(q0, 1, +)	(qF, 1, =)
* qF			



Este obra está bajo una [licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.]

Turing Machine: Example

f	0	1	□
-> q0	(q0, 0, +)	(q1, 1, +)	(qF, 0, =)
q1	(q1, 0, +)	(q0, 1, +)	(qF, 1, =)
* qF			

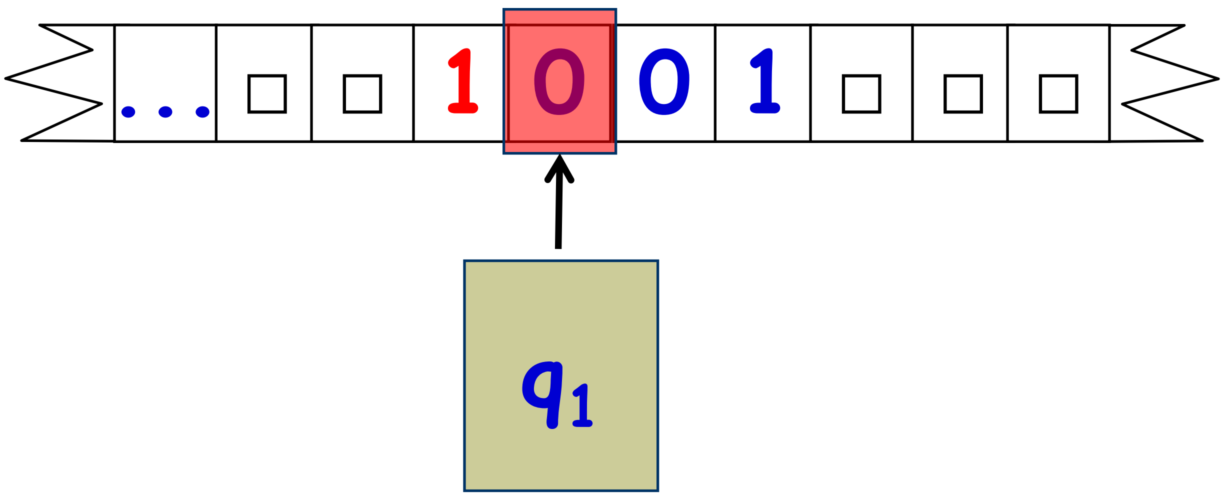


Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Machine: Example

f	0	1	□
-> q0	(q0, 0, +)	(q1, 1, +)	(qF, 0, =)
q1	(q1, 0, +)	(q0, 1, +)	(qF, 1, =)
* qF			

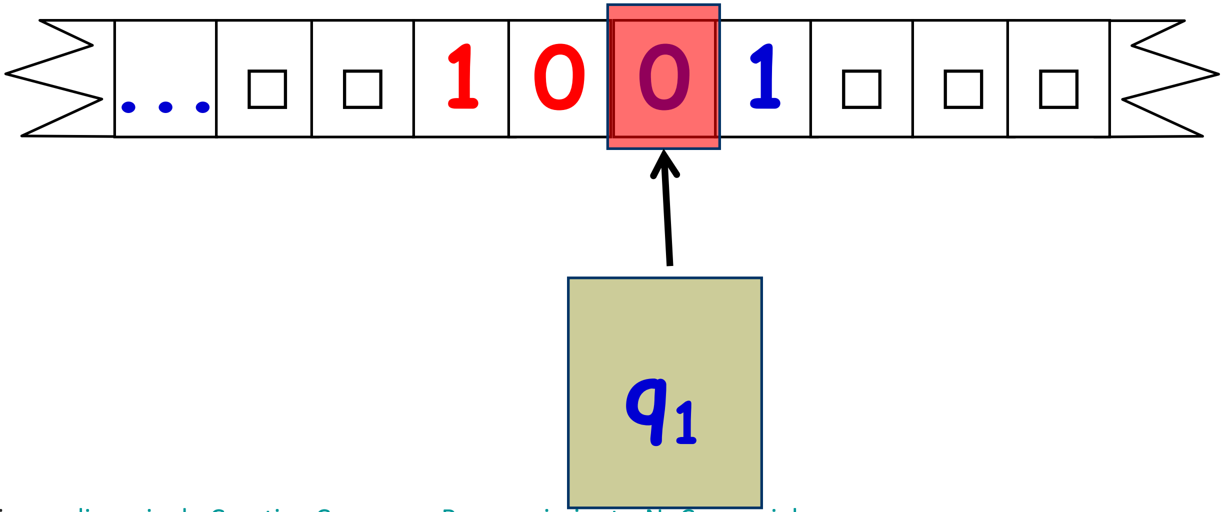


Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Machine: Example

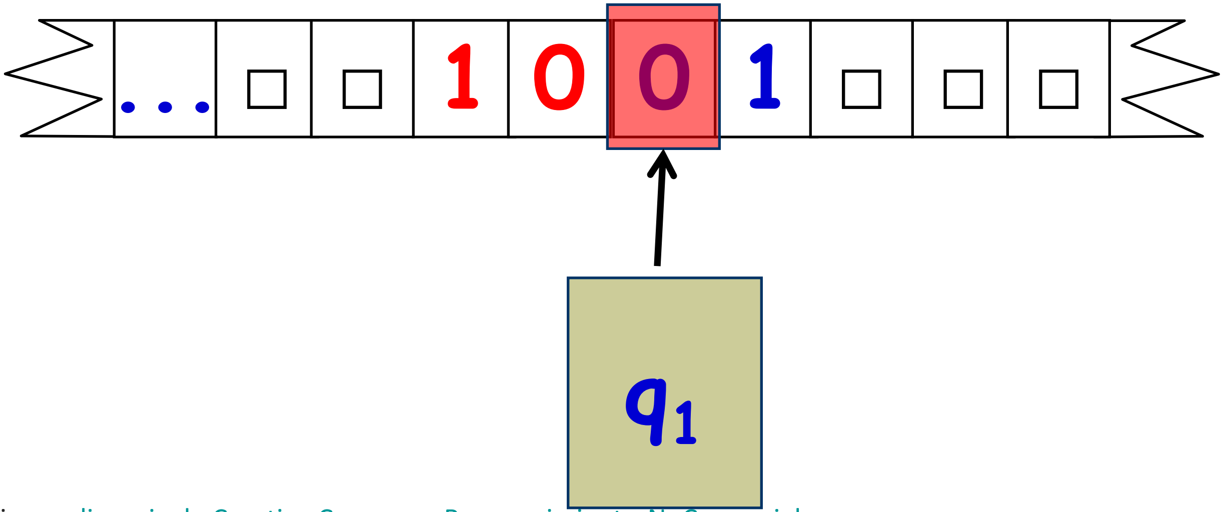
f	0	1	□
-> q0	(q0, 0, +)	(q1, 1, +)	(qF, 0, =)
q1	(q1, 0, +)	(q0, 1, +)	(qF, 1, =)
* qF			



Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Turing Machine: Example

f	0	1	□
-> q0	(q0, 0, +)	(q1, 1, +)	(qF, 0, =)
q1	(q1, 0, +)	(q0, 1, +)	(qF, 1, =)
* qF			

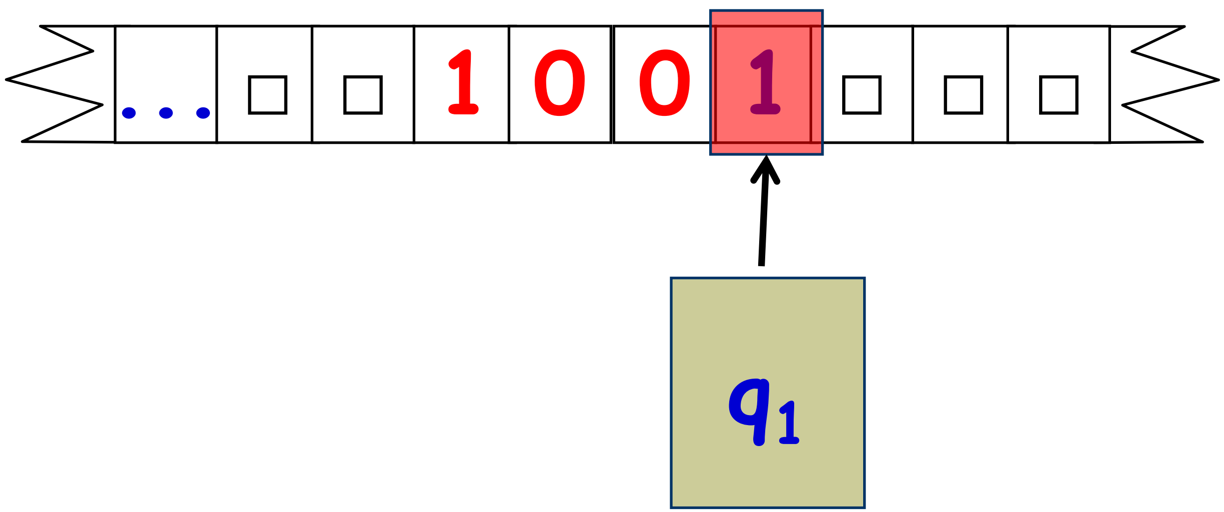


Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Machine: Example

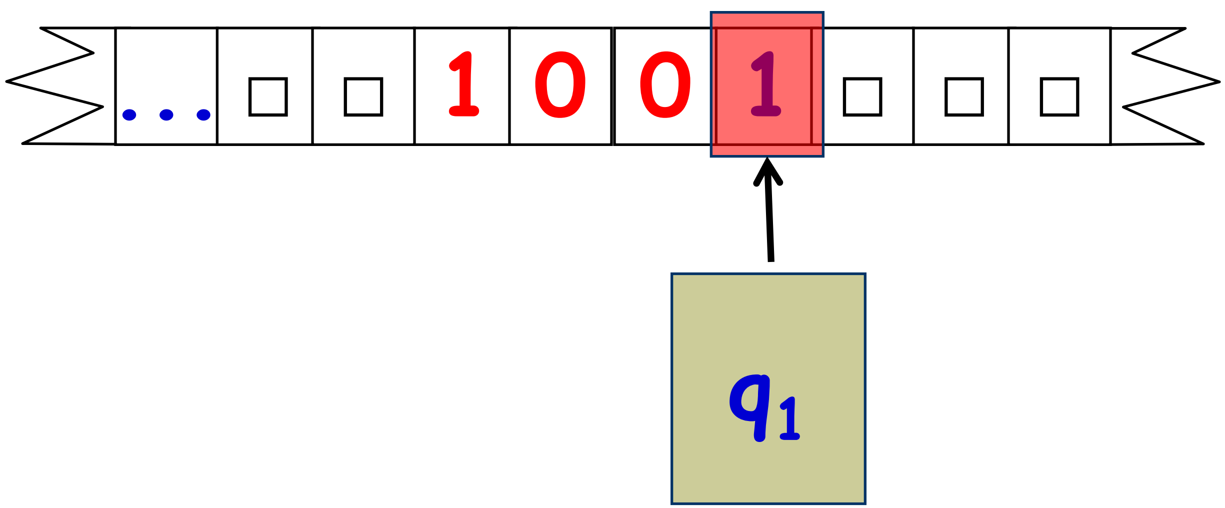
f	0	1	□
-> q0	(q0, 0, +)	(q1, 1, +)	(qF, 0, =)
q1	(q1, 0, +)	(q0, 1, +)	(qF, 1, =)
* qF			



Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Turing Machine: Example

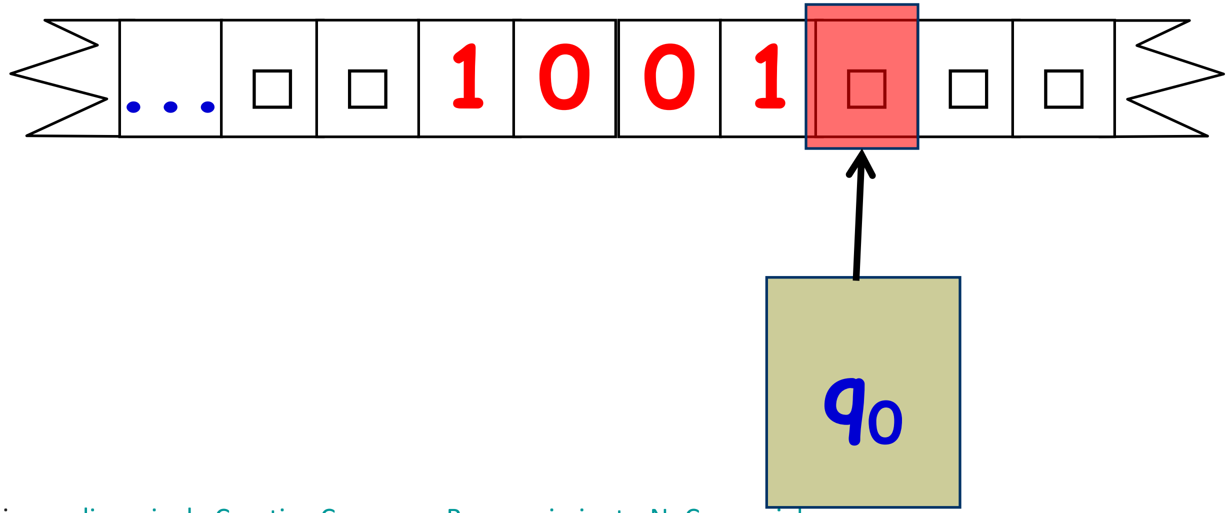
f	0	1	□
-> q0	(q0, 0, +)	(q1, 1, +)	(qF, 0, =)
q1	(q1, 0, +)	(q0, 1, +)	(qF, 1, =)
* qF			



Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Turing Machine: Example

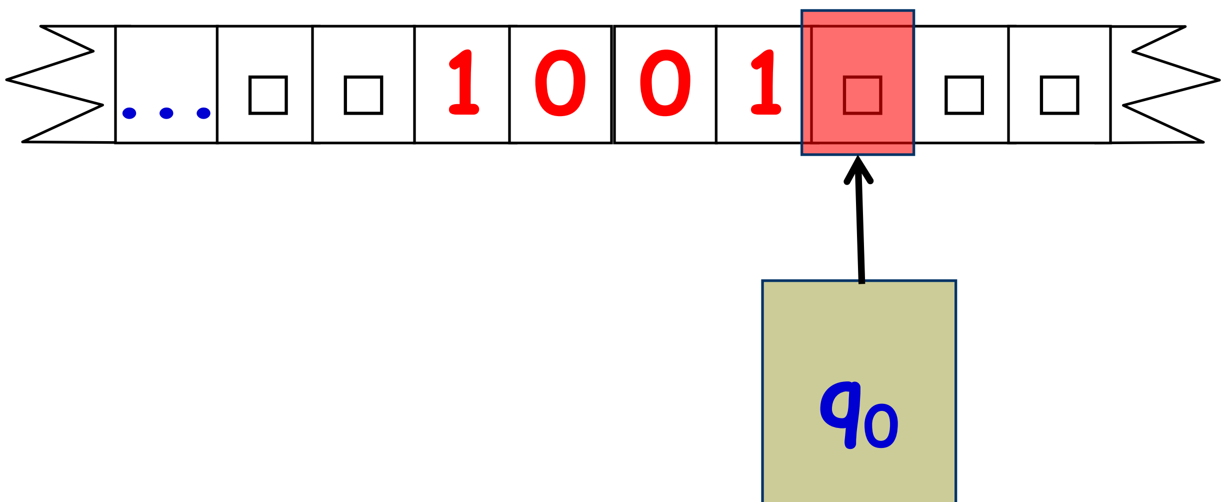
f	0	1	□
-> q0	(q0, 0, +)	(q1, 1, +)	(qF, 0, =)
q1	(q1, 0, +)	(q0, 1, +)	(qF, 1, =)
* qF			



Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Turing Machine: Example

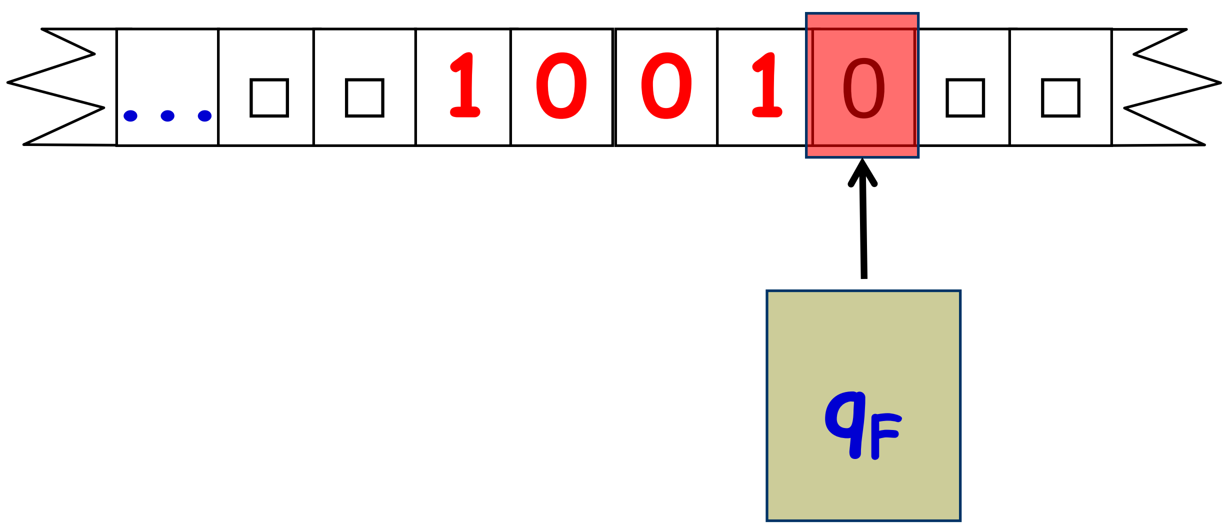
f	0	1	□
-> q0	(q0, 0, +)	(q1, 1, +)	(qF, 0, =)
q1	(q1, 0, +)	(q0, 1, +)	(qF, 1, =)
* qF			



Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Turing Machine: Example

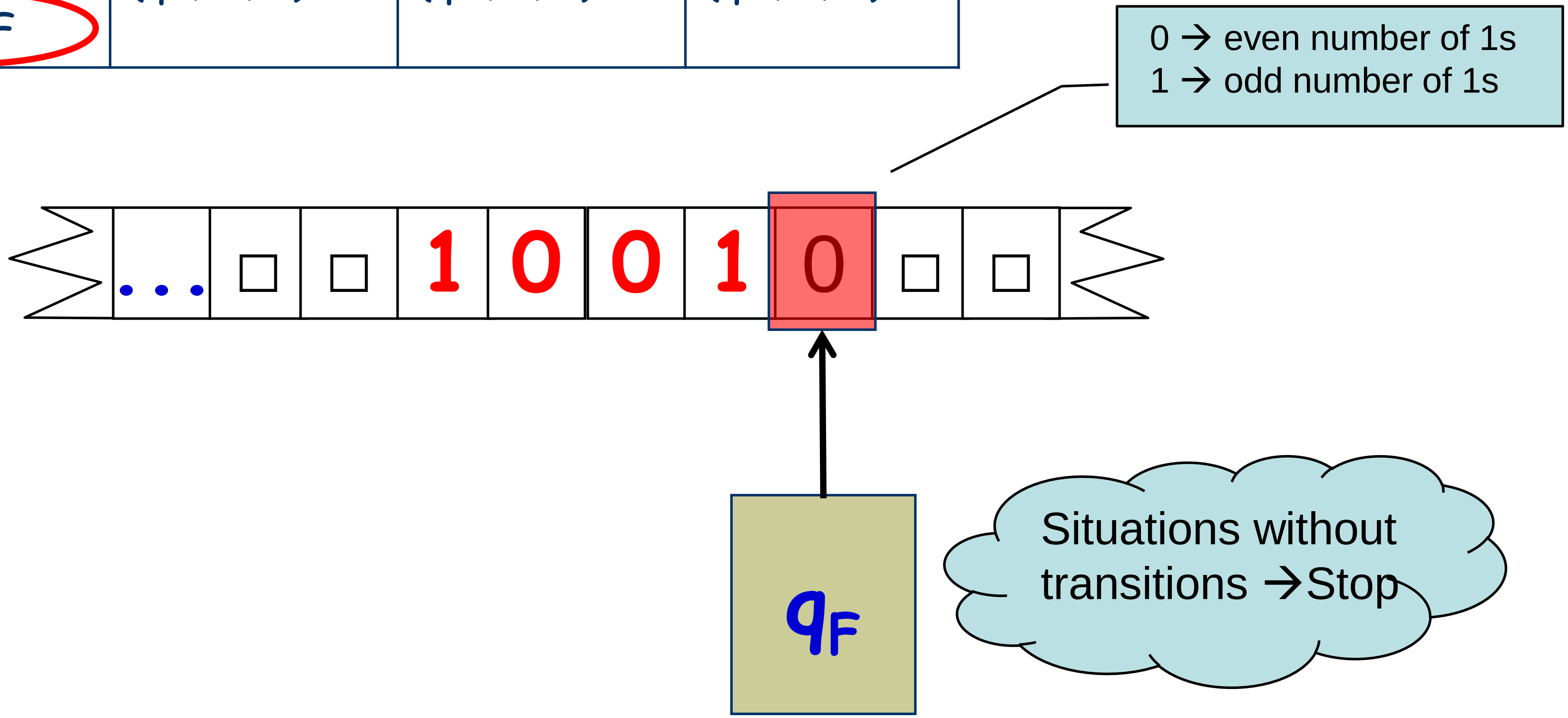
f	0	1	□
-> q0	(q0, 0, +)	(q1, 1, +)	(qF, 0, =)
q1	(q1, 0, +)	(q0, 1, +)	(qF, 1, =)
* qF			



Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Turing Machine: Example

-> q0	(q0, 0, +)	(q1, 1, +)	(qF, 0, =)
q1	(q1, 0, +)	(q0, 1, +)	(qF, 1, =)
* qF			



Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Machine: Graph Representation

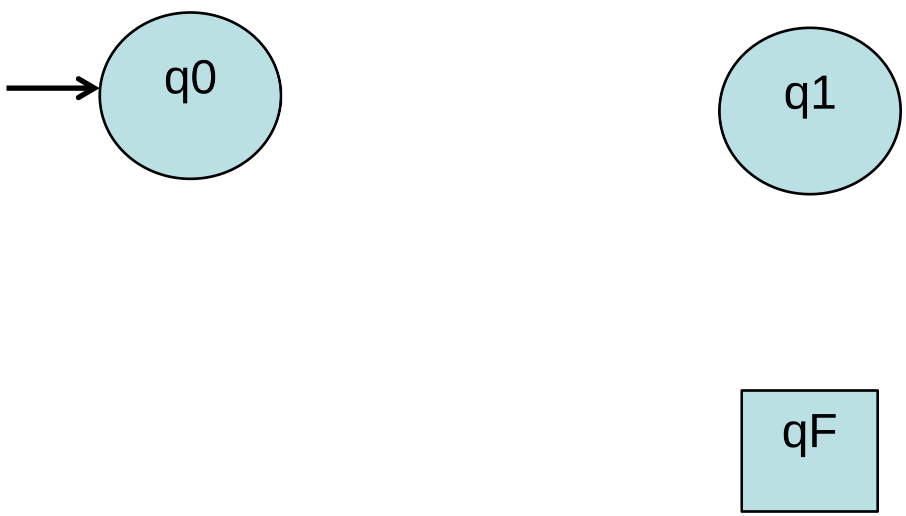
- The transition function can be described using also a diagram with states, i.e. a graph in which:
  - Nodes represent states.
  - Arches represent transitions between states.
  - Each arch is labelled including the requisites and effects of each transition (initial symbol, symbol that is rewritten, and direction to move the input header).

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Machine: Graph Representation

f	0	1	□
-> q0 q1 * qF	(q0, 0, +) (q1, 0, +)	(q1, 1, +) (q0, 1, +)	(qF, 0, =) (qF, 1, =)

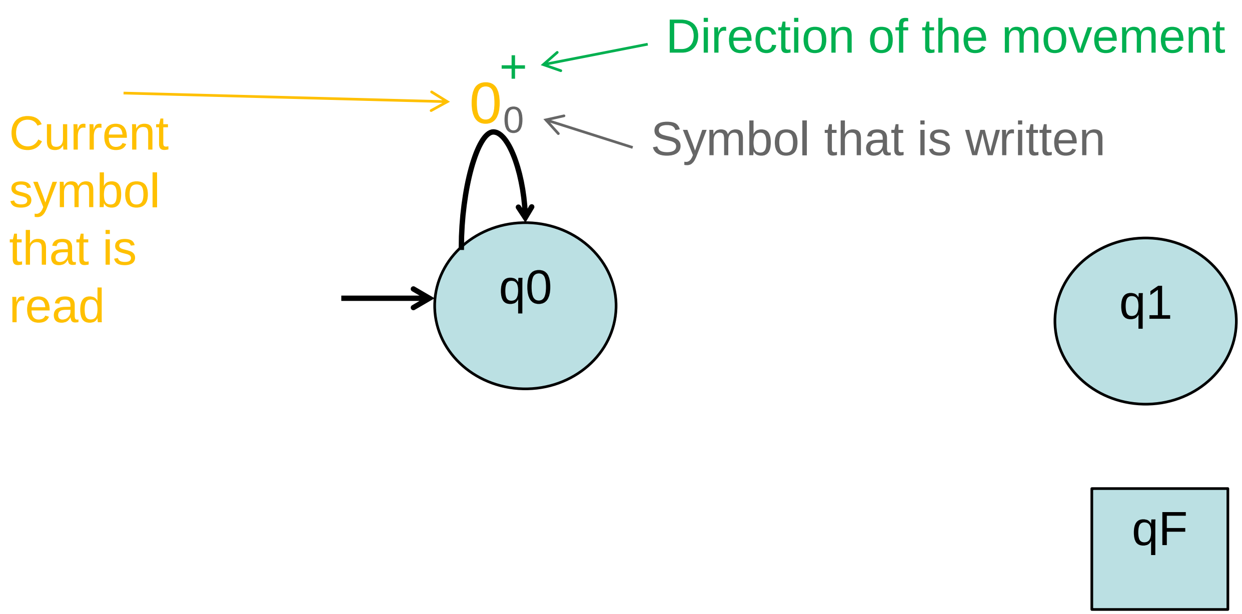


Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Machine: Graph Representation

-> q0 q1 * qF	(q0, 0, +) (q1, 0, +)	(q1, 1, +) (q0, 1, +)	(qF, 0, =) (qF, 1, =)

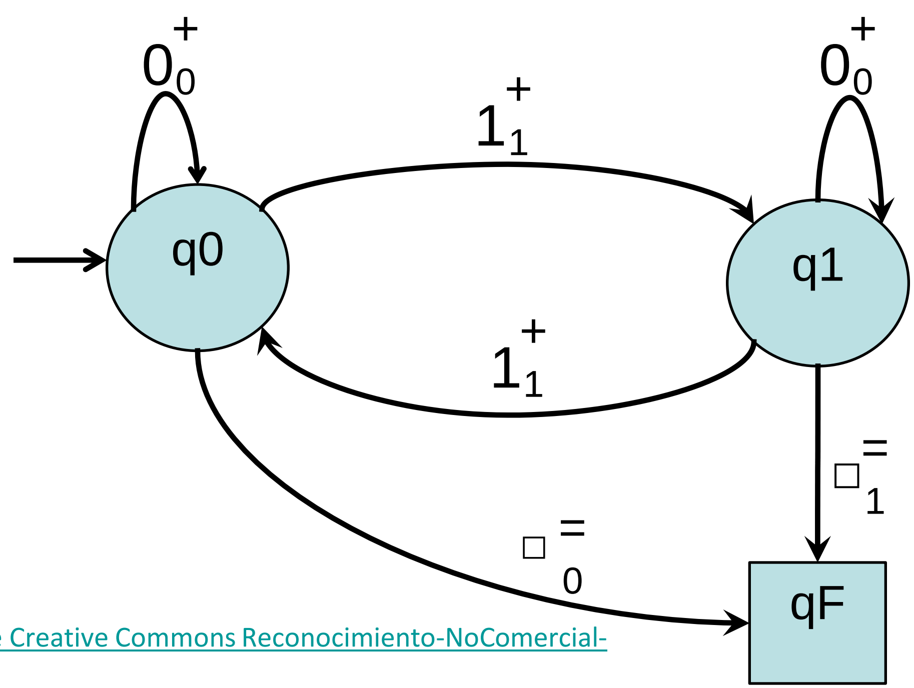


Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Machine: Graph Representation

f	0	1	□
-> q0	(q0, 0, +)	(q1, 1, +)	(qF, 0, =)
q1	(q1, 0, +)	(q0, 1, +)	(qF, 1, =)
* qF			

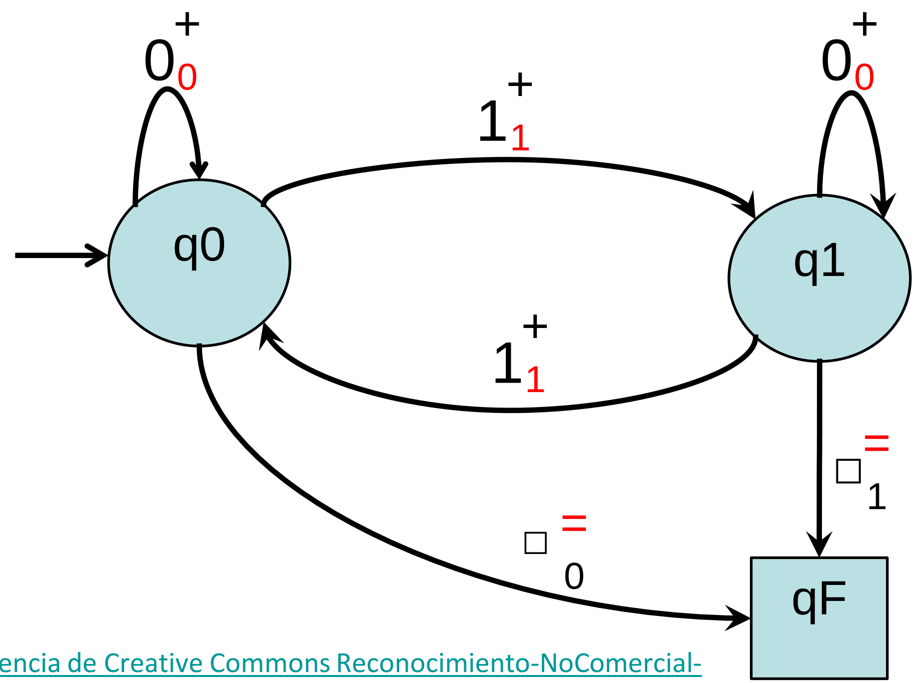


Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Machine: Graph Representation

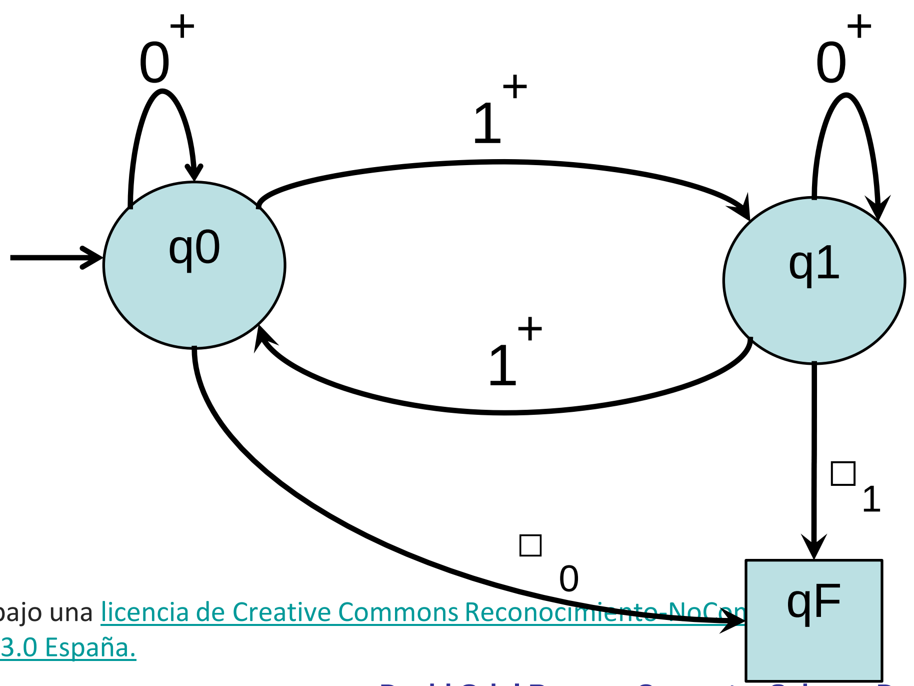
f	0	1	□
-> q0	(q0, 0, +)	(q1, 1, +)	(qF, 0, =)
q1	(q1, 0, +)	(q0, 1, +)	(qF, 1, =)
* qF			



Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Turing Machine: Graph Representation

f	0	1	□
-> q0	(q0, 0, +)	(q1, 1, +)	(qF, 0, =)
q1	(q1, 0, +)	(q0, 1, +)	(qF, 1, =)
* qF			



Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Turing Machine and Languages

- Turing Machine as TRANSDUCER:
  - It modifies the content of the tape,
  - Examples: TM that replaces digits by zeros.

    TM that appends a parity bit to the input.
- Turing Machine as RECOGNIZER:
  - TM that recognizes a language.
  - TM that accepts a language.

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Machine and Languages

- Turing Machine as TRANSDUCER:
  - **Objective: transform the input (→ Provide the result of an operation).**
    - It verifies:
  - If the input is well-formed, it must finish in a final state.
  - If the input is NOT well-formed, it must finish in a nonfinal state (shows an error in the input word).

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Machine and Languages

- Turing Machine as RECOGNIZER
  - **Objective: Decide if the input string is valid or not, following a specific criterion.**
- Two main concepts: RECOGNIZE, ACCEPT
  - A TM **RECOGNIZES** a language L, if for any input in the tape, w, it stops in a final state iff $w \in L$.
  - A TM **ACCEPTS** a language L if, when analyzing a word w, it stops in a final state iff $w \in L$.
    - If the word does not belong to the language, the TM does not need to stop.

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

OUTLINE

- Definition of Turing Machine
- **Variations of Turing Machines**
- Universal Turing Machine
- Additional issues

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Variations of Turing Machines

- We have defined a generic Turing Machine.

- The definition of TM admits many variations, but having the same computation capacity, i.e. **all the variations are equivalent.**

- Two Turing Machines, TM_1 y TM_2 are equivalent **if both carry out the same action for all their inputs**.

  - What does equivalent mean for a Turing Machine?

    ✓ TM **as transducers:** for each possible input, at the beginning of the process, **at the end of this the contents of the tape must be the same.**

    ✓ TM **as recognizers**: both TM **accepts the same words.**

    ✓ If for some input *w*, one TM does not stop, the second one will not stop for such input *w*.

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Variations of Turing Machines

- Using the generic TM it is possible to impose restrictions, without they suppose limitations in the computation capacity.

- These restrictions can be imposed on:

  - **The alphabet of the tape: binary TM.** Γ={0,1}. (~~Not Σ = {0,1} y Γ = {0,1, b}).~~

  - **The structure of the tape: Limited TM.**

    - E.g.: infinite tape only by the right (limited by the left using the input word).

  - **Movements (to write, move and change the state)**

    - E.g.: not to allow that the head remains quiet; not to allow that it writes and changes simultaneously of state; etc.

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Machine with Binary Alphabet

Theorem:

Given a generic TM , there is an equivalent TM with binary alphabet in the tape, Γ = {0,1}

Turing Machine with Binary Alphabet

$M \rightarrow M_{(2}$

IMPORTANT: it is Γ = {0,1} and Σ ⊆ Γ , ~~not Σ = {0,1} and Γ = {0,1, b}~~

Different descriptions in the bibliography

(According to section 2.2.3, Alfonseca 2007)

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Submachines (Subroutines)

- Same concept that functions, methods, procedures (subprograms).
- A TM can, during its execution, invoke to another one.

- **DEFINITION:**
    - A submachine of Turing is a TM that can receive arguments (i.e. symbols) to carry out a specific task specify, and can be invoked with another TM.

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Submachines (Subroutines)

- When a TM M_1 invokes another TM M_2 with an argument σ, the parameter in the definition of the submachine is replaced by the symbol by means it has been invoked: $M_2(\sigma)$.
  - $M_2(\sigma)$ is executed:
    - Using the same tape that M1.
    - From the initial state of M2.
    - With the R/W head of M2 on the same position where it was in M1.

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Submachines (Subroutines)

- To use Turing Submachines does not mean that the computability capacity (calculation) of TM is increased.

- Example:

  - Design a TM (using submachines) that recognizes the words of the language L={$a^n b^n$, $n \geq 0$}

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Submachines (Subroutines)

Example: L={$a^n b^n$, n ≥ 0}

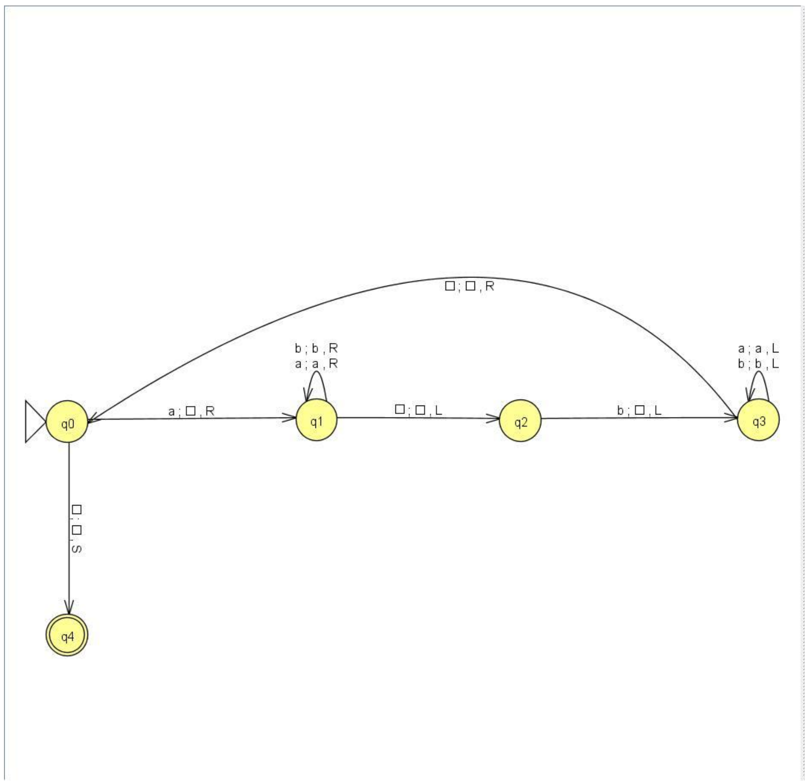


Table Text Size

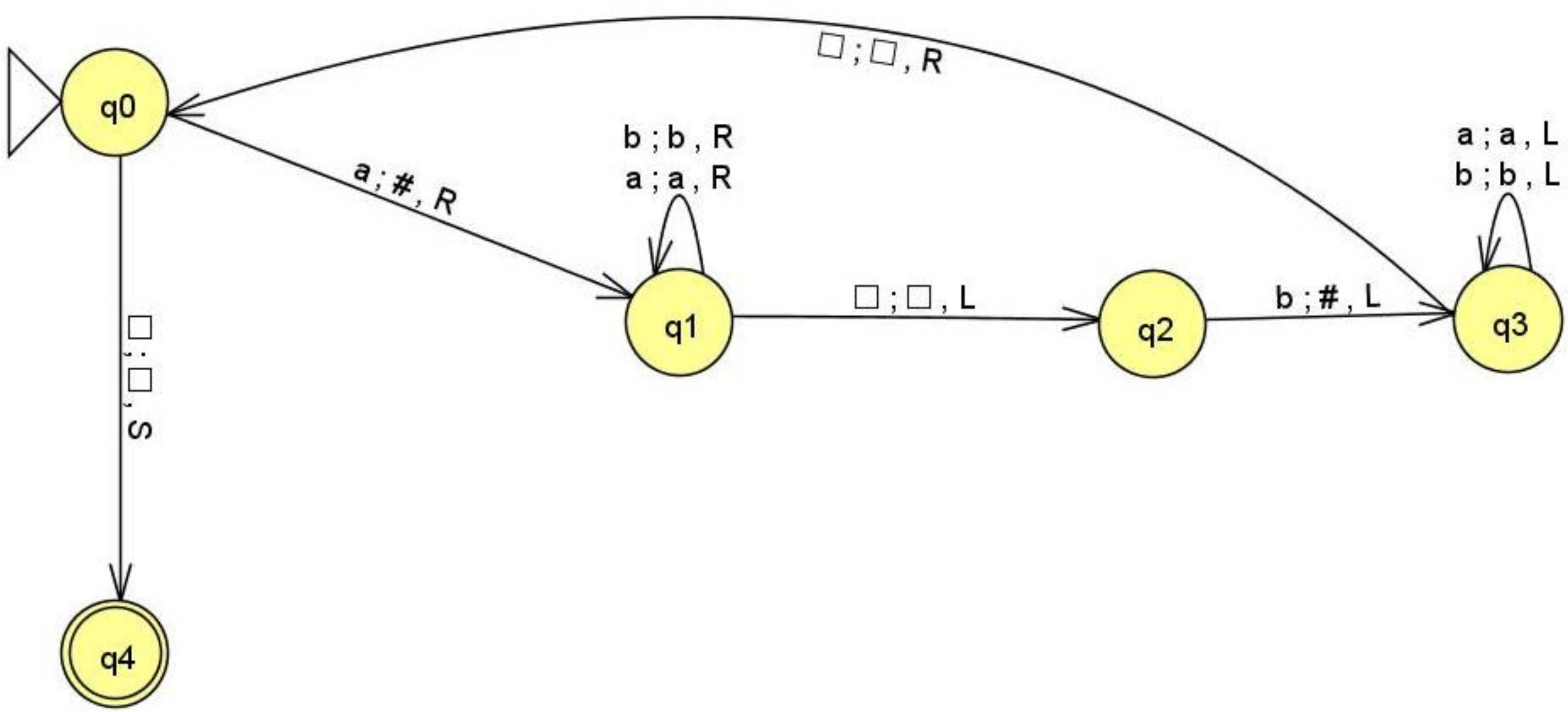
Input	Output	Result
ab		Accept
ba		Reject
aabb		Accept
aab		Reject
abb		Reject
aaaabbbb		Accept

Load Inputs | Run Inputs | Clear | Enter Lambda | View Trace

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-Compartirigual 3.0 España.

Turing Submachines (Subroutines)

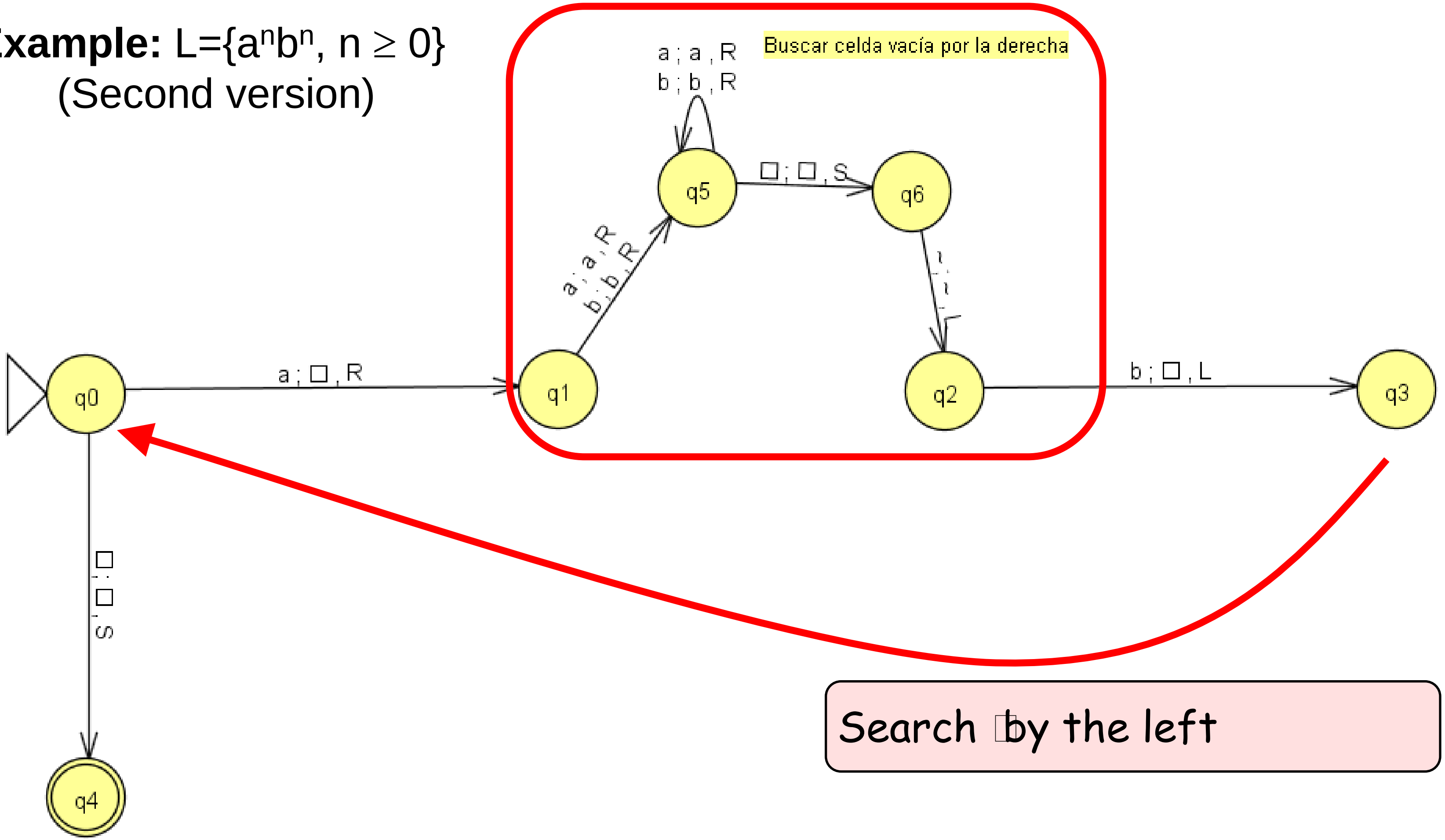
Example: L={$a^n b^n$, n ≥ 0}



Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Turing Submachines (Subroutines)

Example: L={$a^n b^n$, n ≥ 0}
(Second version)



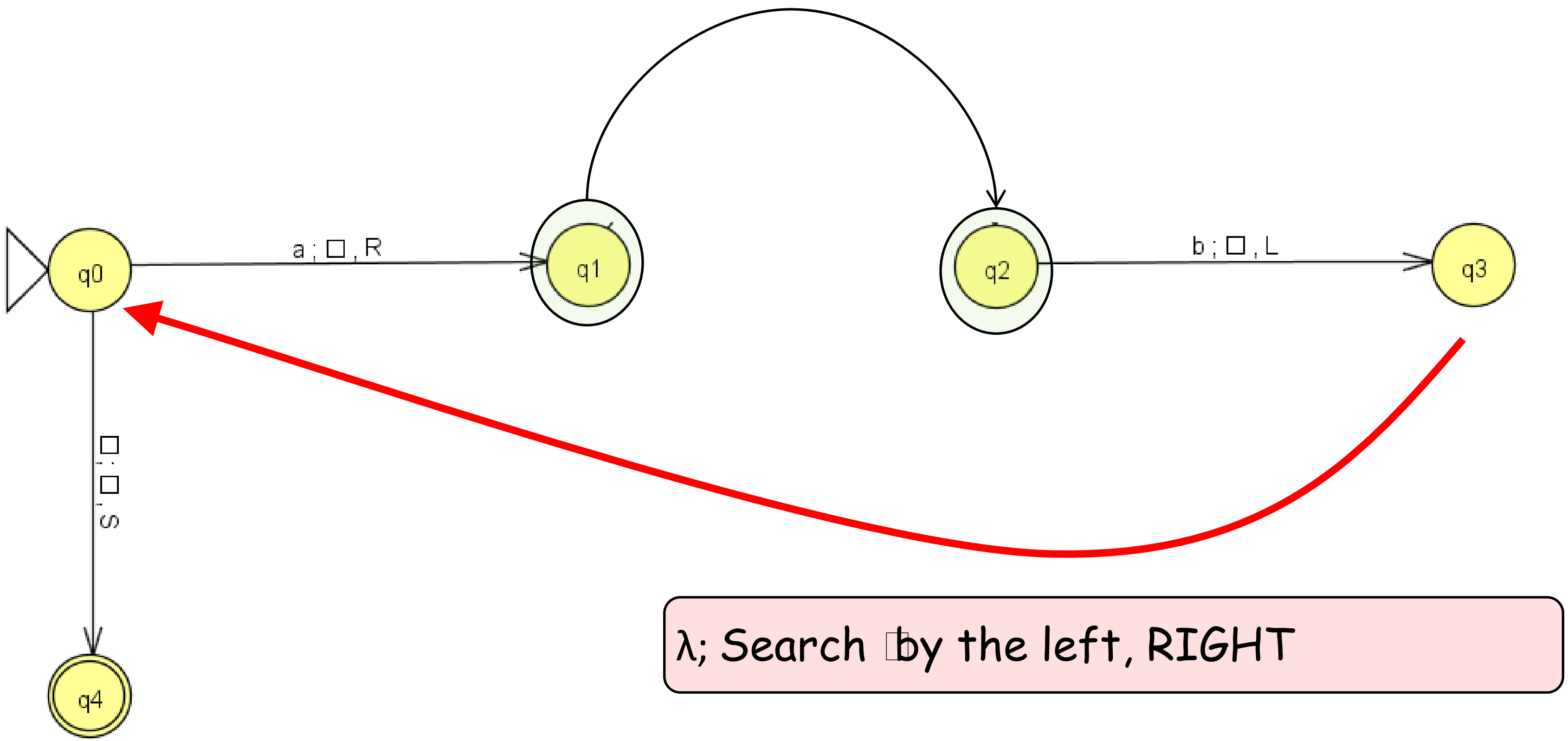
Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Turing Submachines (Subroutines)

Example: L={$a^n b^n$, n ≥ 0}
(using submachines)

λ; Search □ by the right, LEFT

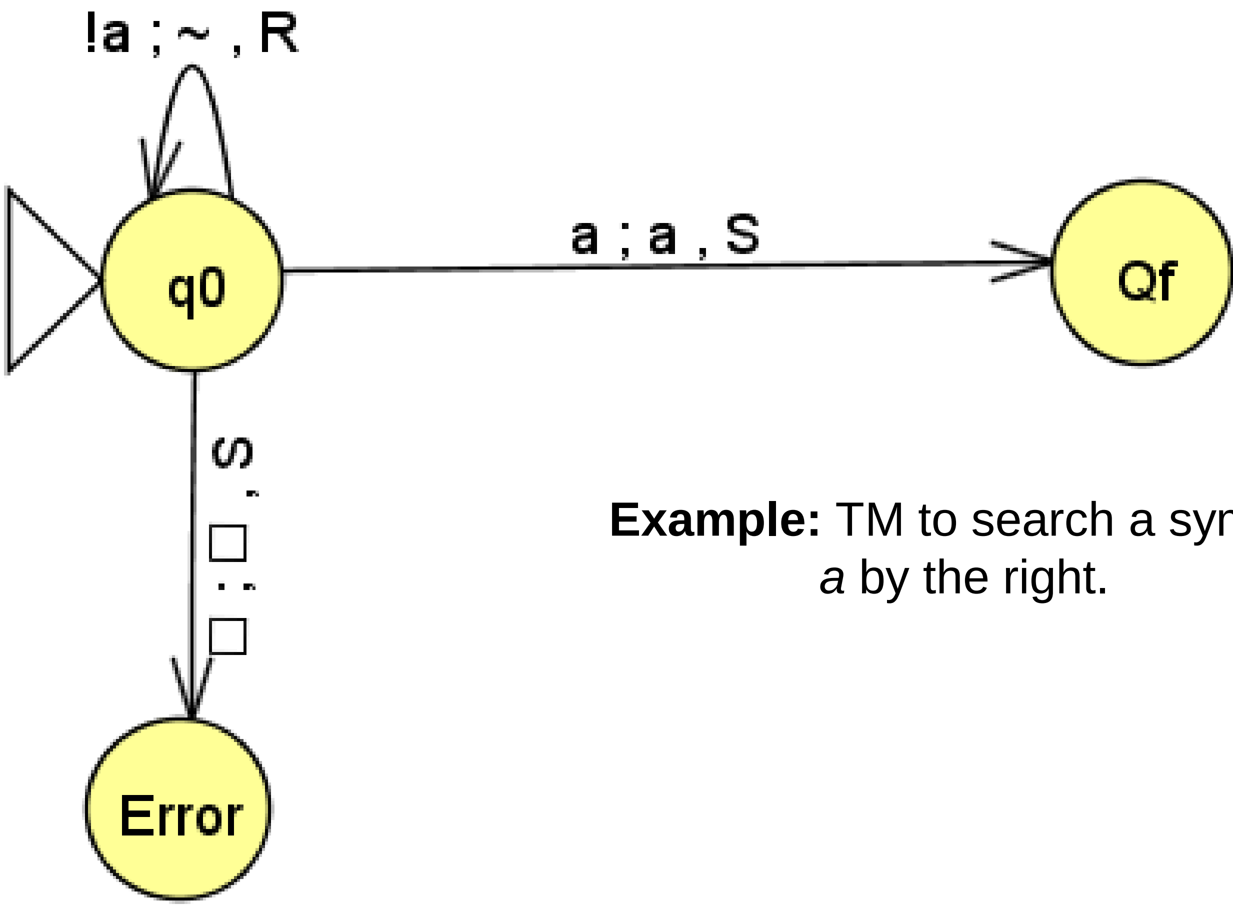


Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Variations of Turing Machines

Turing Submachines (Subroutines)



Example: TM to search a symbol *a* by the right.

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

OUTLINE

- Definition of Turing Machine
- Variations of Turing Machines
- **Universal Turing Machine**
- Additional issues

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Universal Turing Machine

- It is possible to define a TM that **simulates the execution of EVERY** TURING MACHINE.

This machine is called

UNIVERSAL TURING MACHINE (U)

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Universal Turing Machine

- **Universal Turing machine: U**
  - Input:
    - Description of the TM.
    - Input string (word): w.
  - What does U(TM, w) simulate?
    - It simulates the operation of TM, transition by transition.
  - If TM accepts w, then U stops in a final state for that word.
  - If TM does not accept w, U does not stop or it stops in a non-final state for this word.

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Universal Turing Machine

- **Universal Turing Machine: U**
  - It simulates a TM.
  - What does it happen if TM never stops?

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Universal Turing Machine: Undecidability

- HALTING PROBLEM
  - Remember: “*The output of the machine—i.e., the solution to a mathematical query—can be read from the system once the TM has stopped.”*
  - However, in the case of Gödel’s undecidable propositions, the machine would never stop, and this became known as the “halting problem.”
    - It is not possible to decide if, given an input word w, the machine will stop or will be always running.
    - There is not a TM that answers this question.

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

OUTLINE

- Definition of Turing Machine
- Variations of Turing Machines
- Universal Turing Machine
- **Additional issues**

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Design issues

- The solution of the problem is the transition table, but this is not enough to complete an exercise.
- It is essential to think and describe the algorithm before to be implemented, as well as the behavior of each one of the states.
- It is precise to evaluate the machine with a significant set of inputs. There are maybe “difficult” cases that have to be considered.
- Many TM operates on input tapes with a specific configuration, that normally we do not verify.

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

Design issues

- Any solution is not valid, you must think the algorithm very well and verify that it works for every case.
- If TM must operate as if it had “memory”, additional states can be incorporated (in more complicated machines, the input data can be written in specific positions of the tape).
- If a TM must remember positions within the tape, they usually use additional markers.
- If the task can be divided in several sequential subtasks (or subprogram), then do it.

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es