26 Maggio2006

Automi, Trasduttori & Morfologia

Marco Pennacchiotti

pennacchiotti@info.uniroma2.it

Tel. 06 7259 7717

Ing.dell'Informazione, *stanza P1B-03* (nuova ala Ing.Inf, primo piano)

Info

- **Cambio Aula**
  - Giovedì → **Aula 3** (11.30 – 13.15)

- **Download Chaos**
  - http://ai-nlp.info.uniroma2.it/external/chaosproject/protectedDownload/download.html

- **Progetto**
  - Formare gruppi 3-5 persone
  - Inviare e-mail per iscriversi entro **giovedì 1 Giugno**
  - http://ai-nlp.info.uniroma2.it/pennacchiotti/teaching/

Programma

- **Breve introduzione all'NLP**
  - Linguaggi Naturali e Linguaggi Formali
  - Complessità
- **Morfologia**
  - *Teoria:* Morfologia del Linguaggio Naturale
  - *Strumenti:* Automi e Trasduttori
  - *Analisi Morfologica:* con automi e trasduttori
- **Part of Speech Tagging**
  - *Teoria:* Le classi morfologiche
  - *Strumenti a Analisi:* modelli a regole e statistici
- **Sintassi**
  - *Teoria:* Sintassi del Linguaggio Naturale
  - *Strumenti:* CFG
  - *Analisi Sintattica:* parsing top-down, bottom-up, Early
- **Semantica**
  - Lexical Semantics
  - Sentence Semantics

Info

Obiettivi dell'NLP

L' Elaborazione del Linguaggio Naturale (*Natural Language Processing, NLP*) si prefigge come obiettivi principali:

- costruzione di modelli e e strumenti informatici in grado di eseguire specifici task riguardanti il Linguaggio Naturale:
  - Permettere la comunicazione *uomo – macchina*
  - Migliorare la comunicazione *uomo – uomo*
  - Elaborare e manipolare oggetti linguistici

PERCHE' E' IMPORTANTE L' NLP ?

- Sempre maggiore quantità di conoscenza condivisa in testi in Linguaggio Naturale *machine readable (ES: il Web)*
- Necessità di un'interazione più diretta uomo-macchina (*ES: agenti intelligenti)*

Livelli di analisi del Linguaggio Naturale

I **sistemi di NLP** possono operare a diversi livelli di analisi, ognuno dei quali richiede una specifica conoscenza linguistica .

- **FONETICA:** studio dei suoni linguistici
- **MORFOLOGIA:** studio delle componenti significative di una parola
- **SINTASSI:** studio delle strutture relazionali tra le parole
- **SEMANTICA:** studio del significato
- **PRAGMATICA:** studio di come il linguaggio è usato per raggiungere obiettivi
- **ANALISI DEL DISCORSO:** studio di unità linguistiche complesse

Una architettura per L'NLP può portare a termine uno o più livelli di analisi, generalmente in cascata

Linguaggio Naturale e *Linguaggi Formali*

- **Cos'è il Linguaggio Naturale ?**
  - Strumento di comunicazione tra persone;
    - Fatti, idee e conoscenze sul mondo esterno ed interiore
    - Emozioni
    - Ordini
  - E' ambiguo! (*"La vecchia porta la sbarra"*)
- **Cos'è un Linguaggio Formale ?**

Dato un insieme di simboli Σ detto alfabeto, un linguaggio formale è un sottoinsieme di tutte le possibili concatenazioni dei simboli:

$$L \subseteq \Sigma^*$$

Un linguaggio formale non è ambiguo (una concatenazione di simboli ha una interpretazione univoca) ed esprime le sue regole in maniera canonica

Un elaboratore può riconoscere e generare solo Linguaggi Formali, attraverso l'utilizzo di modelli e algoritmi

Linguaggi Formali: complessità

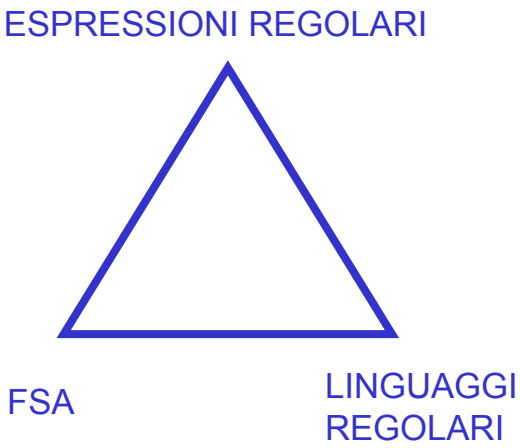
- Le grammatiche sono modelli dichiarativi
- I corrispondenti modelli procedurali sono:

- **Type 0 Grammars - Unrestricted**
  Turing Machine
- **Type 1 Grammars - Context-Sensitive**
  Turing Machine
- **Type 2 Grammars - Context-Free**
  Push-down automaton
- **Type 3 Grammars - Regular**
  Finite State Automaton (FSA)

Automi a Stati Finiti (*FSA*)

Un automa a stati finti è un automa in grado di *riconoscere* o di *generare* una sequenza di simboli (*stringa*).

Formalmente: Un FSA è un grafo orientato i cui nodi sono detti stati e i cui archi sono detti transizioni



Caratteristiche principali:

- molto efficienti (tipo 3 nella ger. di Chomsky)
- facili da implementare
- Ogni FSA implementa una *espressione regolare*
- Ogni espressione regolare descrive un FSA
- Ogni FSA descrive un *linguaggio regolare*

Utilizzi principali in linguistica:

- Riconoscimento morfologico
- Fonetica
- Text-to-Speech

Espressioni regolari

Due parole…

- **Espressione regolare:** notazione algebrica per descrivere un insieme di stringhe
  - Una Espressione Regolare descrive un FSA
  - Un FSA implementa un'espressione regolare

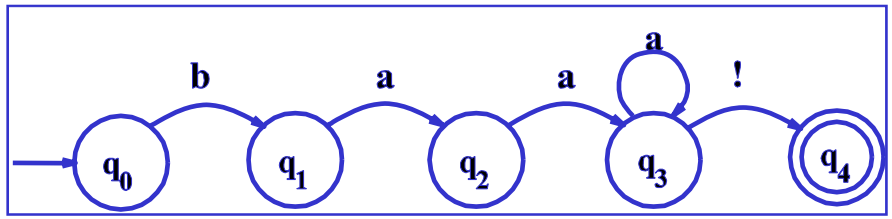
- *Operatori base:*
  - * → zero o più occorrenze del carattere precedente
  - + → una o più occorrenze del carattere precedente
  - ? → zero o una occorreza del carattere precedente
  - [a|A] → disgiunzione di simboli
  - [a-A] → range di simboli

- *Esempi:*
  - /a*/ → L={0,a,aa,aaa}
  - /[ab]+/ → L={a,b,aa,bb,ab,ba,…}
  - *altri esempi sul libro….*

FSA: definizione formale

Un FSA è definito dai seguenti parametri:

- *Q* : un insieme finito di N stati $q_0 \ldots q_N$
- *Σ* : un alfabeto finito di simboli
- q_0 : lo stato iniziale
- *F* : un insieme di stati finali $F \subseteq Q$
- *δ(q,i)* : funzione di transizione tra stati che restituisce un nuovo stato a partire da un dato stato e un simbolo in input



	Input		
State	b	a	!
0	1	0	0
1	0	2	0
2	0	3	0
3	0	3	4
4:	0	0	0

Un FSA può essere anche rapprentato attraverso una *state-transition table*

FSA

1. Disegnare (se esistono) gli FSA che riconoscono/generano i seguenti linguaggi regolari:

   La= {ac,abc,abbc,abbbc, …}

   Lb= {ac,abb}

   Lc= {ac,acdc,acdcdc, …}

   Ld= {ac,ab,acdc,abdb,abdc,acdb,acdcdc, …}

2. Scrivere l'espressione regolare che descrive l'FSA
3. Scrivere le transition table relative agli FSA

ESERCIZIO 1

FSA

1. **Scrivere un FSA che riconosca espressioni "monetarie" del tipo:**

   "uno euro", "due euro", "venti tre euro", "venti uno ero dieci cent", "trenta due euro trenta quattre centesimi", "due cent"….

   Il cui vocabolario sia Σ={uno, due, tre,…dieci,venti,…,euro,cent}

2. **Scrivere la relativa state-transition table**

3. **Scrivere l'espressione regolare associata all'FSA**

ESERCIZIO 2

FSA

1. **Scrivere un FSA che riconosca sintagmi nominali per l'Inglese, ovvero:**

   - nomi propri (es. *"John"*)

   - nomi comuni preceduti da articoli e eventualmente da un numero variabile di aggettivi (es. *"the new yellow table"*)

   - combinazione di sintagmi dei due punti precedenti preceduti da preposizoni (es. *"the new yellow table of John"*)

2. **Scrivere la relativa state-transition table**

ESERCIZIO 3

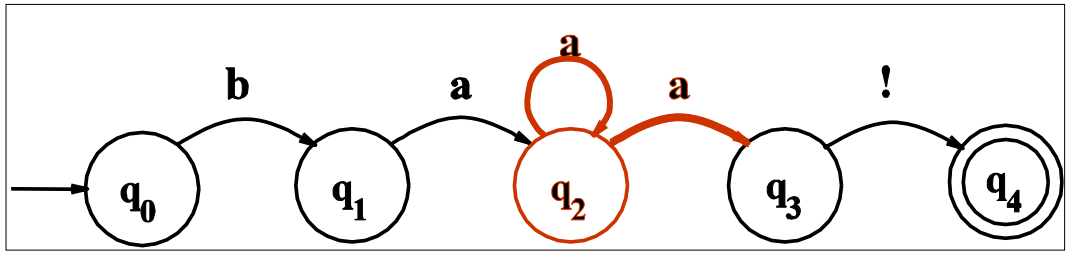
Sommario

Strumenti per la Morfologia

- **Automi a stati finiti (FSA)**
  - FSA deterministici
  - **FSA non-deterministici (NFSA)**
  - Introduzione alla Morfologia
  - FSA e Morfologia: *riconoscimento*
- **Trasduttori a stati finiti (FST)**
  - Cosa sono
  - FST e Morfologia: *parsing*

FSA non-deterministici (NFSA)

Un automa è detto non-deterministico se ha due archi uguali uscenti dallo stesso stato.



Quindi:

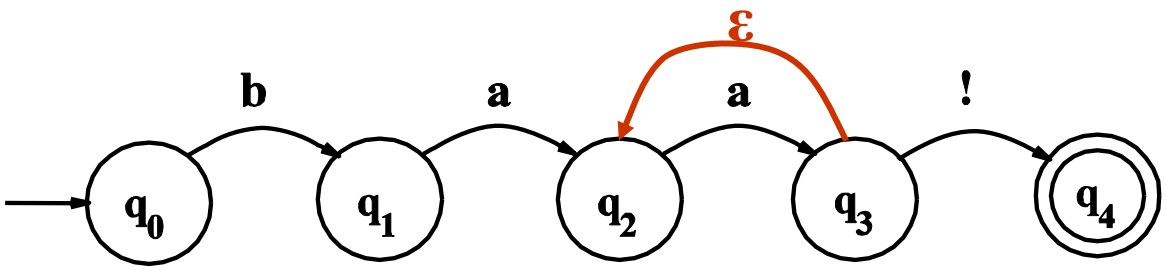
- *Deterministico* vuol dire che ad ogni stato può essere presa una sola decisione
- *Non-Deterministico* vuol dire che ad ogni stato si può scegliere tra più decisioni

Equivalenza tra FSA e NFSA

- Un NFSA può essere sempre convertito in un FSA *equivalente* (che definisce cioè lo stesso linguaggio)
- NFSA e FSA hanno quindi lo stesso potere di riconoscimento/generazione
- L'FSA equivalente di un NFSA ha sempre più stati dell'NFSA

FSA non-deterministici (NFSA)

Un tipo particolare di non-determinismo è quello causato dalla presenza di ε-transizioni (o *jump arcs)*



Una ε-transizione corrisponde ad un passaggio di stato che non influenza la stringa in esame:

- *in riconoscimento:* non viene letto il simbolo corrente della stringa

in generazione: non viene prodotto alcun simbolo

FSA non-deterministici: ricerca

Riconoscimento: negli stati non-deterministici l'FSA può seguire *strade* diverse, ovvero prendere decisioni *errate.* In tal caso deve essere in grado di:

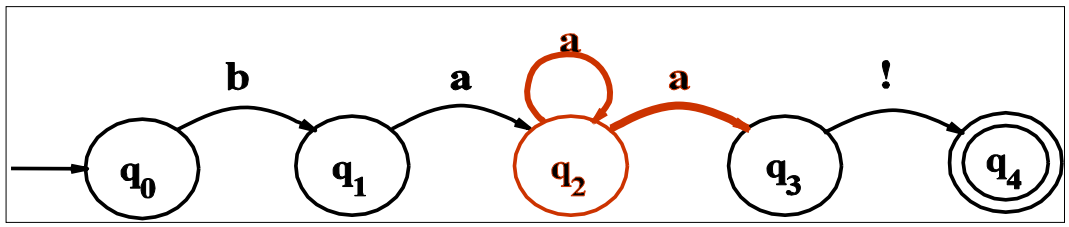
- Riconoscere la soluzione errata;
- Cercare altre soluzioni prendendo strade diverse;
- Ricordare quali sono le strade diverse

L'automa deve quindi effettuare una ricerca nello spazio delle soluzioni (*state-space search*)

Ad ogni *bivio* (choice point) devono quindi essere memorizzate in una *agenda* tutte le coppie di stati alternativi e la posizione nella stringa dopo la transizione δ (*search-states*)

STATO CORRENTE

q2, [b,a,a,a,a]

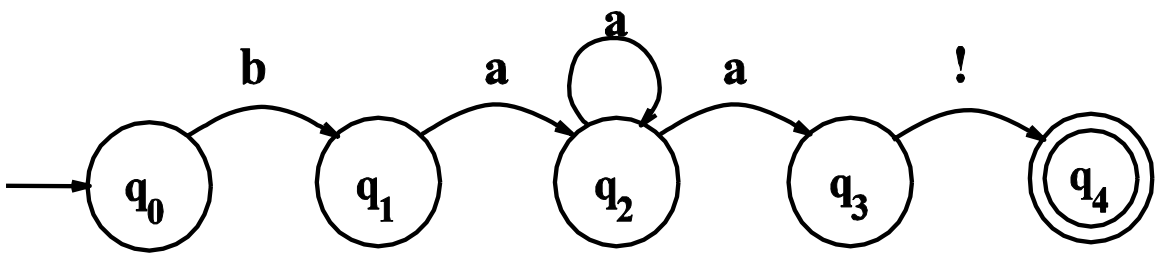


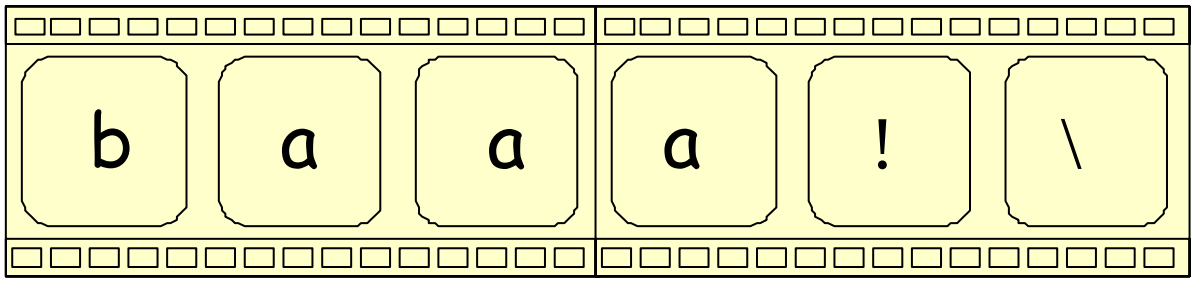
SEARCH STATES

q3, [b,a,a,a,a]

q2, [b,a,a,a,a]

Ricerca in NFSA: esempio

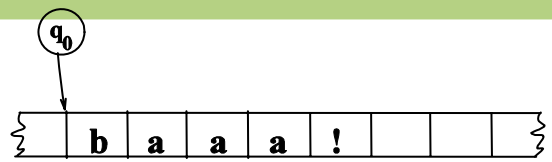




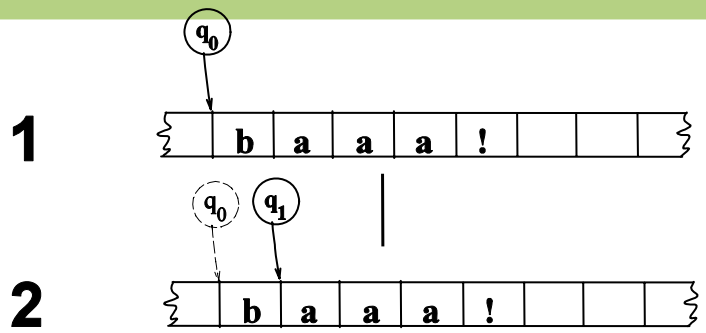
q_0 q_1 q_2 q_2 q_3 q_4

Ricerca in Profondità NFSA: esempio

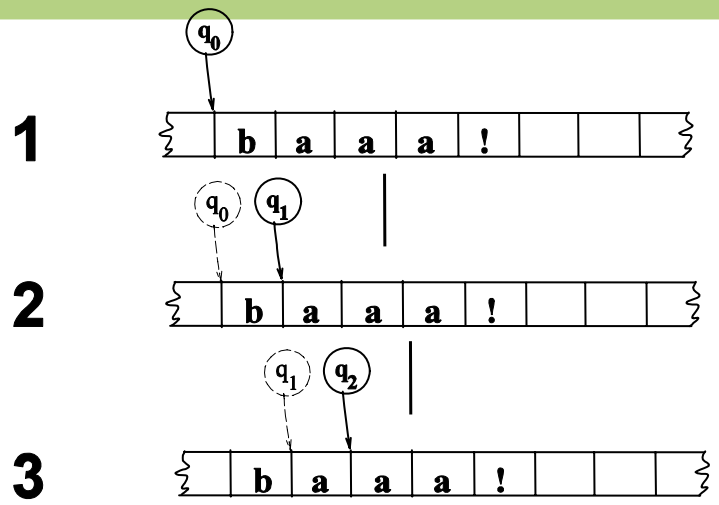
1



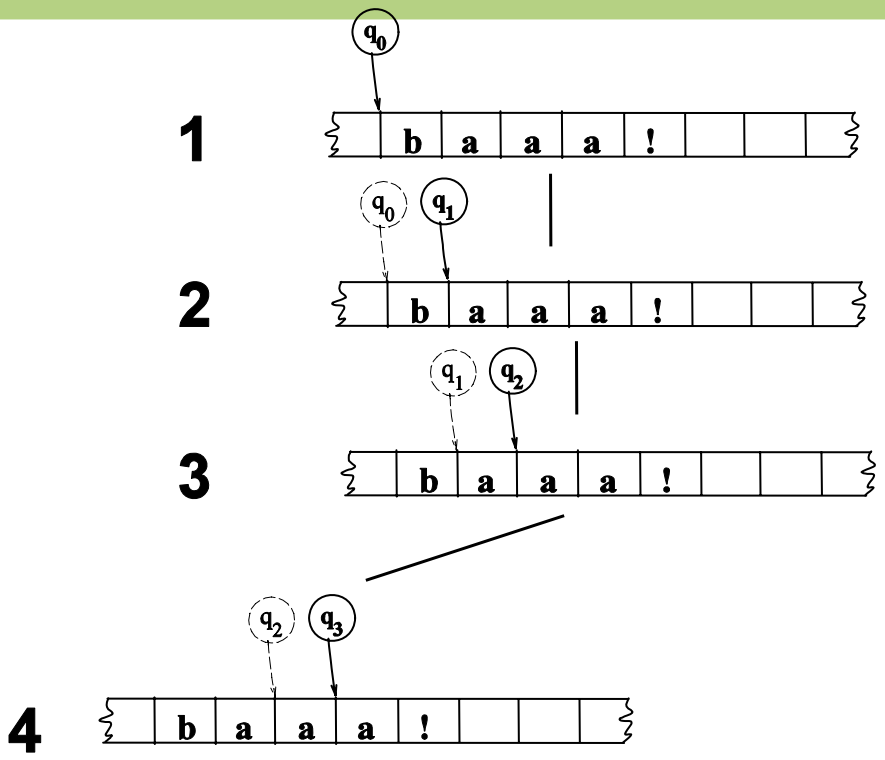
Ricerca in Profondità NFSA: esempio



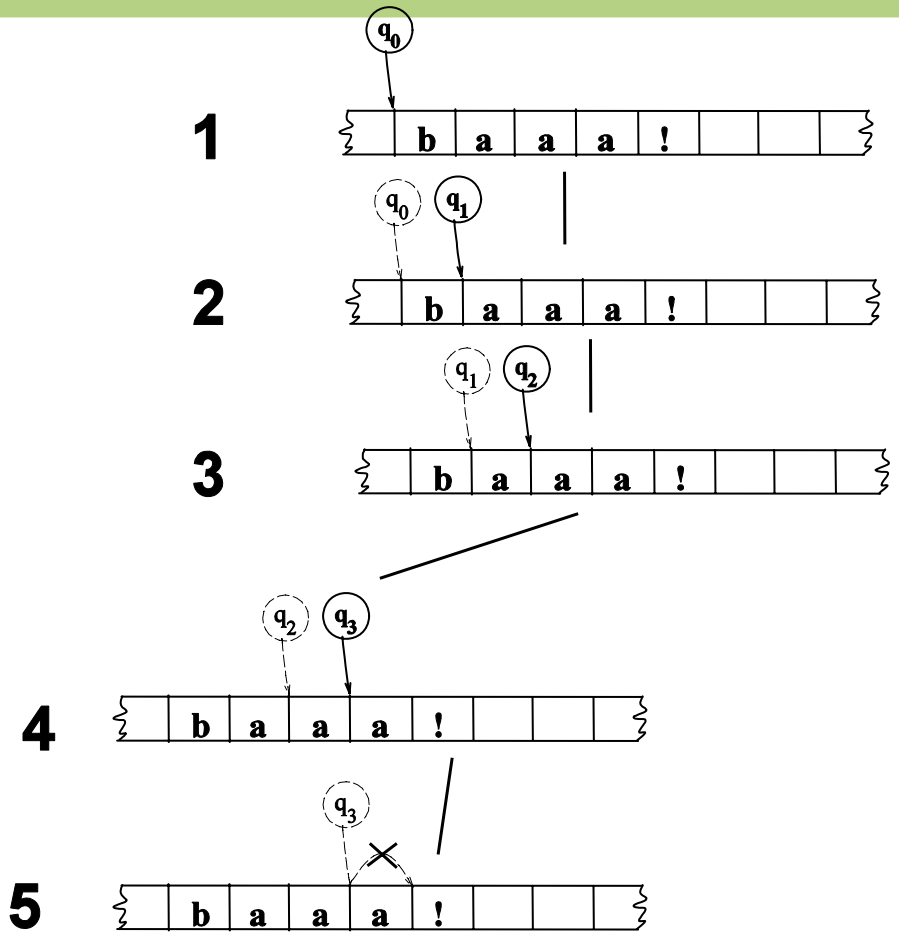
Ricerca in Profondità NFSA: esempio



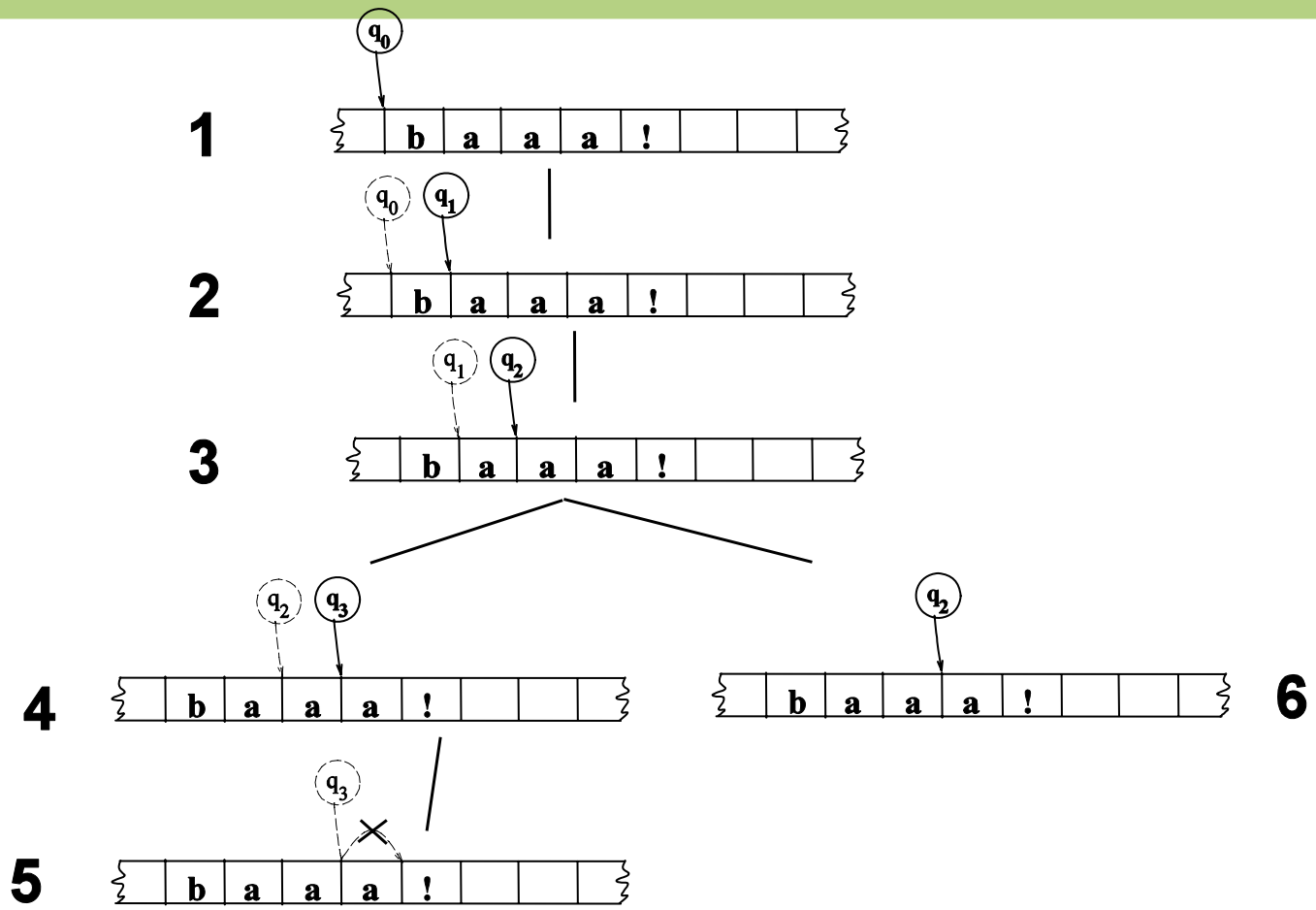
Ricerca in Profondità NFSA: esempio



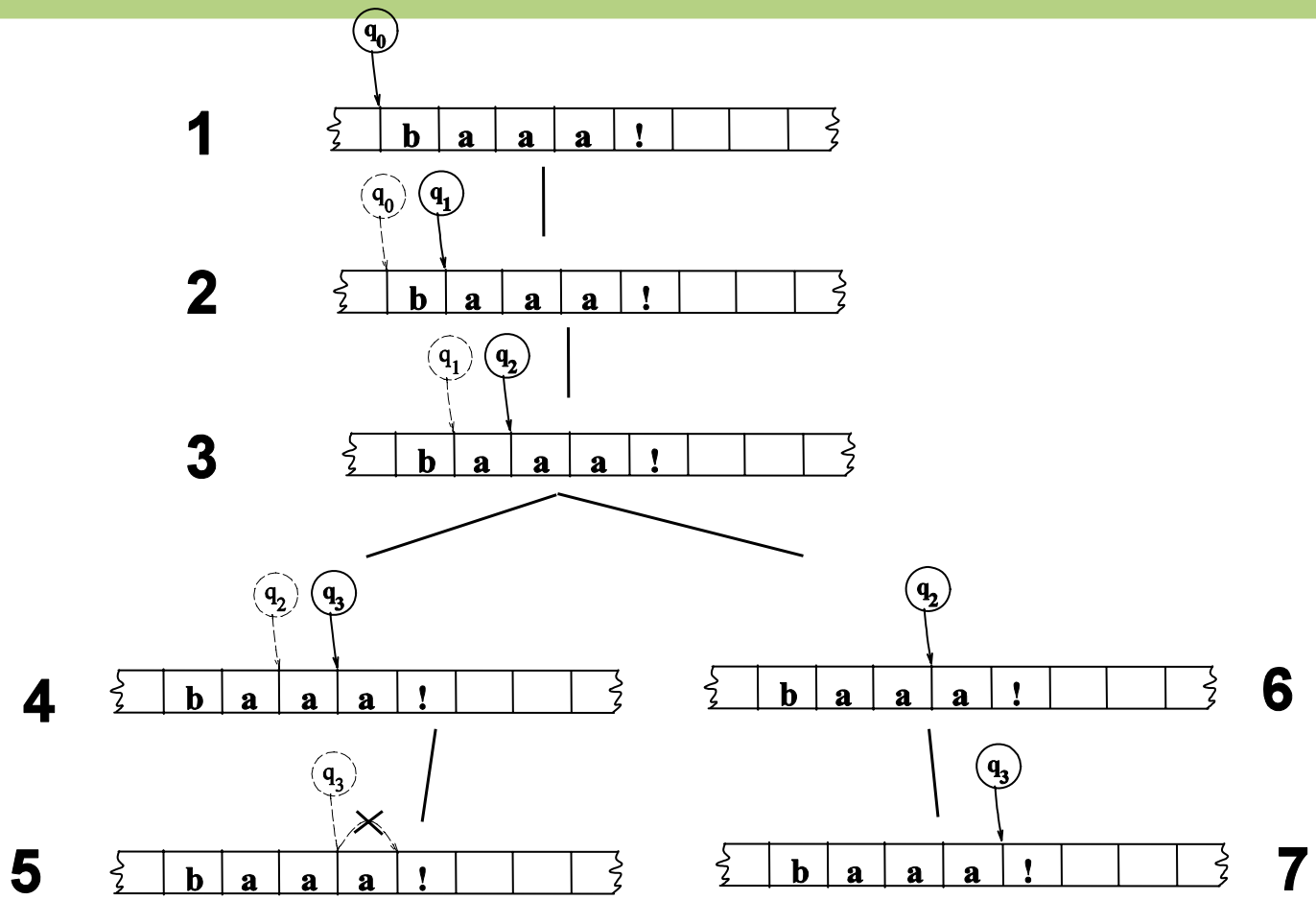
Ricerca in Profondità NFSA: esempio



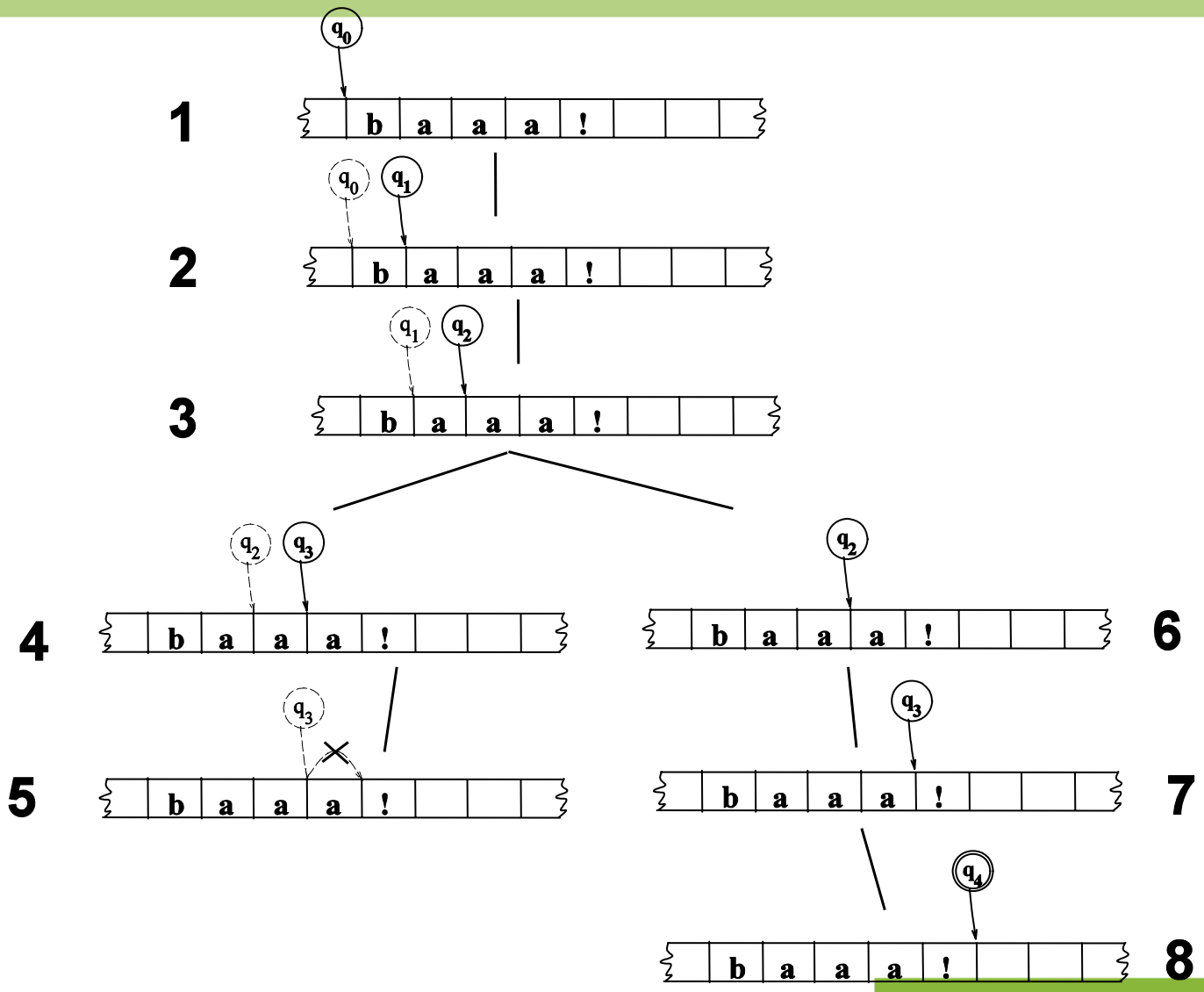
Ricerca in Profondità NFSA: esempio



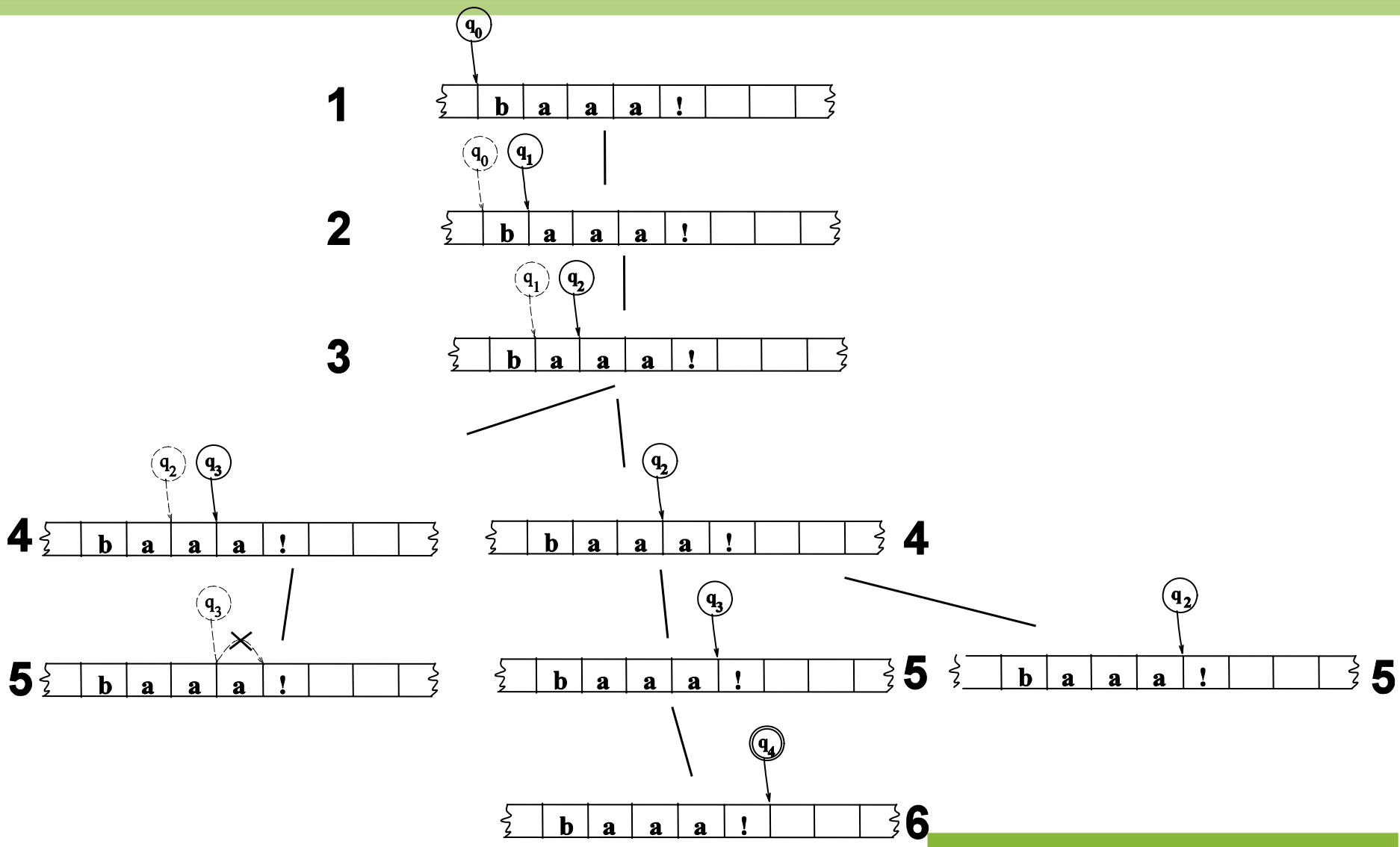
Ricerca in Profondità NFSA: esempio



Ricerca in Profondità NFSA: esempio



Ricerca in Ampiezza NFSA: esempio



Ricerca in NFSA: algoritmo di ricerca

```
function ND-RECOGNIZE(tape, machine) returns accept or reject

  agenda ← |(Initial state of machine, beginning of tape)|
  current-search-state ← NEXT(agenda)
  loop
    if ACCEPT-STATE?(current-search-state) returns true then
       return accept
    else
       agenda ← agenda ■ GENERATE-NEW-STATES(current-search-state)
    if agenda is empty then
       return reject
    else
       current-search-state ← NEXT(agenda)
  end
```

Sommario

Strumenti per la Morfologia

- **Automi a stati finiti (FSA)**
  - FSA deterministici
  - FSA non-deterministici (NFSA)
  - **Introduzione alla Morfologia**
  - FSA e Morfologia: *riconoscimento*
- **Trasduttori a stati finiti (FST)**
  - Cosa sono
  - FST e Morfologia: *parsing*

Morfologia: *definizioni*

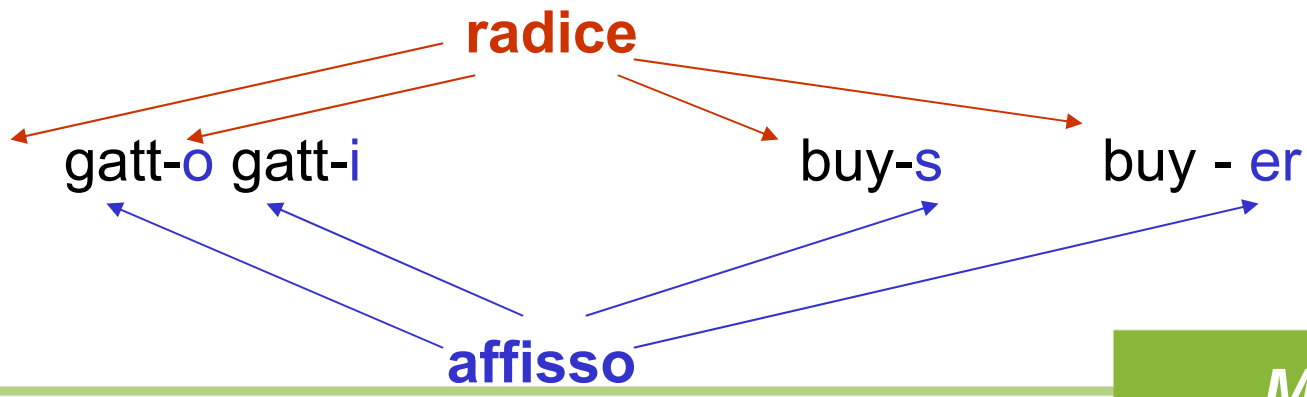
La morfologia (*morphology*) è lo studio di come le parole sono costruite a partire da unità atomiche dette *morfemi*.

I morfemi (*morphemes*) sono le più piccole unità linguistiche che possiedono un significato. Possono essere divisi in due classi:

- Radice (*stem*)→ il morfema che dà il significato principale alla parola

- Affisso (*affix*)→ particelle apposte alla radice che ne completano il significato e la funzione grammaticale

ESEMPIO



Morfologia: *definizioni*

La morfologia può essere divisa in due parti principali

- Inflectional Morphology: combinazione di una radice con un affisso che risulta in una parola (*forma flessa*) della *stessa classe* (nome, verbo, aggettivo, ecc..) con una funzione grammaticale specifica

  - *cat* (nome sing) → *cat-**s*** (nome plur)
  - *cut* (verbo base) → *cut-**ting*** (verbo progressivo)

- Derivational Morphology: combinazione di una radice con un affisso che risulta in una parola di una *classe diversa.* Il significato della nuova parola non è facilmente prevedibile

  - *trasporto* (nome) → *trasport-**abile*** (aggettivo)
  - *computerize* (verbo) → *computeriz-ation* (nome)

Quante morfologie ?

Ogni linguaggio naturale ha una sua morfologia (affissi, regole, ecc.)

Due classi principali:

- Concatenative morphology: una parola è composta da morfemi concatenati insieme

  - Italiano, Inglese: *gatt-**o**, cat-**s**, buy-**er***

- Non-concatenative morphology: le parole sono composte in maniera complessa

  - Ebraico: *l**a**m**a**d* (radice: *lmd*; affissi: *a-a* pass.rem.attivo)

Lingue agglutinanti: le parole sono formate da molti affissi

- Turco: ***uygarlaştiramadiklarimizdanmişsinizcasina***
  ("comportarsi come se fossi tra quelli che non possono civilizzare")

Morfologia inglese

Caratteristiche:

- Concatenative morphology
- Non-agglutinative (al più 4 o 5 affissi)
- Una parola può avere più affissi:
  - *Prefissi:* ***un****-certain*
  - *Suffissi:* *eat-****s***
  - Combinazioni: ***un****-clear-****ly***
- Inflectional morphology: semplice, applicata solo a *nomi, verbi* e *aggettivi*
- Derivational Morphology: complessa

Inflectional Morphology

NOMI:

Due solo inflessioni:

- Plurale: *cat → cat-s* *thrush → thrush-es*
- Possessivo: *dog → dog's* *children → childrens'*

VERBI:

Quattro forme morfologiche:

- stem: *walk*
- *s* form: *walk → walk-s*
- past form: *walk → walked*
- *ing* form: *walk → walking*

- **Irregolari:** **(ca. 250)** Parole che non seguono le regole morfologiche (Esempio:*mouse → mice go → goes, going, went*). La maggior parte dei nomi e verbi inglesi sono regolari

- La classe dei verbi regolari è **produttiva** : una nuova parola della lingua è automaticamente inclusa nella classe (Esempio: *fax → faxes,faxing, faxed*)

Derivational Morphology

NOMINALIZZAZIONE (*verbo, aggettivo → nome)*

-ation	computerize	computerization
-ee	appoint	appointee
-er	kill	killer
-ness	fuzzy	fuzziness

nome, verbo →aggettivo

-al	Computation	Computational
-able	Embrace	Embraceable
-less	Clue	Clueless

Sommario

Strumenti per la Morfologia

- **Automi a stati finiti (FSA)**
  - FSA deterministici
  - FSA non-deterministici (NFSA)
  - Introduzione alla Morfologia
  - **FSA e Morfologia: *riconoscimento***
- **Trasduttori a stati finiti (FST)**
  - Cosa sono
  - FST e Morfologia: *parsing*

A cosa serve l'analisi morfologica automatica?

- *Stemming* in Information Retrieval
  - Data una parola della query, cercare le pagine che contengano anche le sue forme flesse
- *Spell Checking*
  - Riconoscere quali forme flesse sono ammissibili in una lingua e quali no (ad esempio *gatt-o* e *are-gatt*)
- Traduzione Automatica
  - Ricondurre parole diverse a una stessa radice e quindi alla stessa traduzione (ad esempio *amatore*, *amare* → *love*)

Quali strumenti usare ?

- *Lessico esteso*

  - Un lessico (lista di parole) che contiene tutte le parole della lingua in tutte le forme flesse
  - Spreco di spazio e non è *produttivo*!

- *Lessico ridotto + Automi*

  - La morfologia è generalmente produttiva (gran parte delle parole segue le regole morfologiche per formare le forme flesse)
  - Conviene quindi utilizzare:
    - Lessico contenente solo radici e affissi (ed eventualmente irregolarità)
    - Implementazione delle regole morfologiche in un dispositivo
    - FSA sono semplici dispositivi per implementare tali regole

FSA: *riconoscimento*

Un FSA può essere utilizzato per riconoscere se una parola è ammissibile in una lingua

Cosa serve:

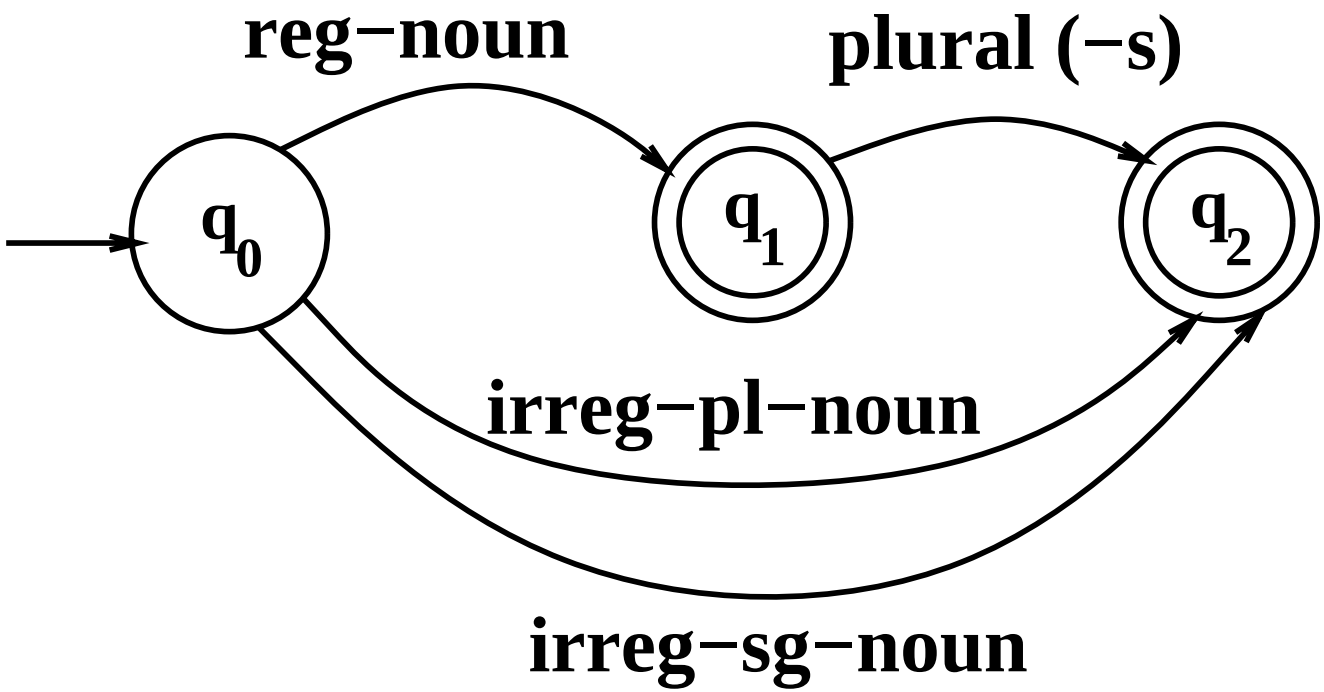
1. Lessico: lista di radici ed affissi della lingua
   - Esempio: [cat,dog,cut,go,…,-s,-ed,-ation,-able,…,un-,dis-]

2. Regole Morfologiche (*morphotactics*): le regole di costruzione dei morfemi
   - Esempio: Plurale inglese: radice + -s

3. Regole Ortografiche: cambiamenti che occorrono in una parola quando due morfemi si combinano
   - Esempio: *city* → *cities*

NOMI: *regole morfologiche*

FSA per modellare l'inflessione plurale per nomi regolari ed irregolari

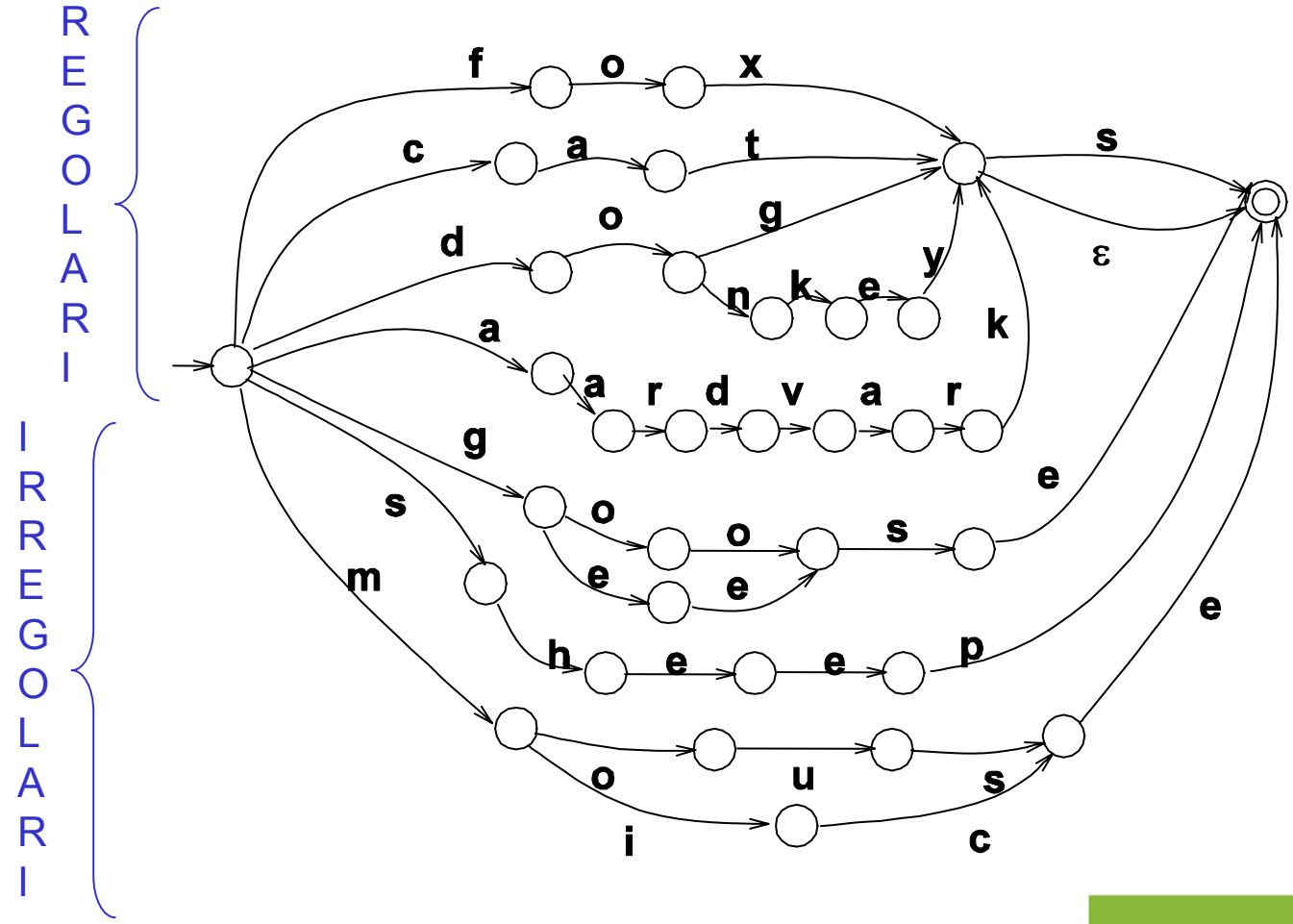


Come modellare i nomi (regolari ed irregolari) nell'FSA?

Ovvero: Come si può integrare il lessico?

NOMI: *regole morfologiche + lessico*

Integrazione del lessico dei nomi regolari ed irregolari



FSA per la morfologia

1. **Scrivere un FSA che riconosca la morfologia derivazionale degli aggettivi inglesi, ovvero:**
   - ***Un aggettivo può avere come prefisso negante* "un-"**
   - ***Un aggettivo può avere forma comparativa, superlativa e avverbiale (rispettivamente i suffissi* –er,-est,-ly*)***

2. **Aggiungere all'FSA il seguente fatto:**
   - ***Esistono alcuni aggettivi "irregolari" che non possono prendere* "un-" *e* "-ly" (es: *big, cool*)**

3. **Integrare il lessico: regolari: *"clear, happy"*, irregolari: *"big,cool"***

ESERCIZIO 4

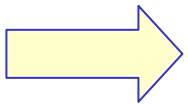
Sommario

Strumenti per la Morfologia

- **Automi a stati finiti (FSA)**
  - FSA deterministici
  - FSA non-deterministici (NFSA)
  - Introduzione alla Morfologia
  - FSA e Morfologia: *riconoscimento*
- **Trasduttori a stati finiti (FST)**
  - Cosa sono
  - FST e Morfologia: *parsing*

Dal Riconoscimento al Parsing

RICONOSCIMENTO : indica se una data parola in input è morfologicamente corretta o no (ad esempio *gatti* è corretta, *gattare* è scorretta)



FSA

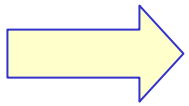
PARSING/GENERAZIONE :

- parsing: produce un'analisi morfologica della parola in input: data la parola in input viene restituita la sua struttura

$$cats \rightarrow cat\ +N\ +PL$$

- generazione: data una struttura morfologica in input, produce una *forma superficiale* (parola)

$$cat\ +N\ +PL \rightarrow cats$$



FST

Trasduttori a Stati Finiti (*FST*)

I *Trasduttori* sono automi a stati finiti con due nastri *A* e *B*

Ad es, può *leggere* da un nastro (ad es. "*cats*") e *scrivere* sull'altro ("*cat* + *N* + *PL*")

Quattro modalità di utilizzo dell' FST:

- riconoscitore: riceve in input una coppia di stringhe su *A* e *B,* e restituisce *accept* se essa appartiene al linguaggio delle coppie
  - *cats, cat+N+PL → accept*
- produttore: restituisce coppie di stringhe appartenenti al linguaggio su *A* e *B*
  - Output: tutte le parole del lessico con la loro struttura
- traduttore: riceve in input una stringa su *A (o B*) e ne restituisce un'altra su *B* (o *A*)
  - *cats → cat+N+PL (PARSING)*
  - *cat+N+PL → cats (GENERAZIONE)*

FST: definizione formale

Un FST è definito dai seguenti parametri:

- *Q* : un insieme finito di N stati $q_0 \ldots q_N$

- *Σ* : un alfabeto finito di simboli complessi. Ogni simbolo complesso è una coppia di simboli ***i:o*** appartenenti rispettivamente agli alfabeti **I** e **O** ($\Sigma \subseteq I \times O$)

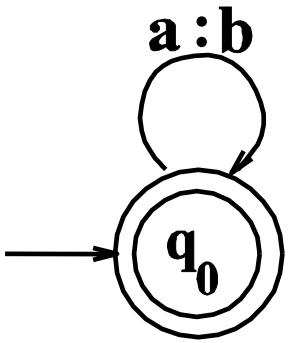
- q_0 : lo stato iniziale

- *F* : un insieme di stati finali $F \subseteq Q$

- *δ(q,i:o)* : funzione di transizione tra stati che restituisce un nuovo stato a partire da un dato stato e un simbolo complesso in input

---

ESEMPIO



- *riconosce* tutte le coppie di stringhe in cui una ha tutte *a* e l'altra uno stesso numero di *b*

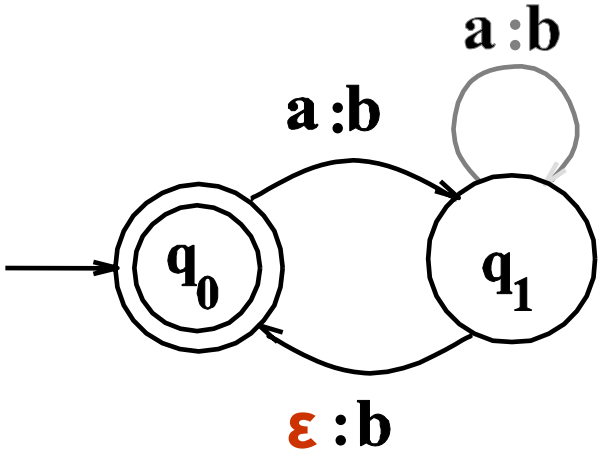
- *produce* stringhe di *a* su un nastro e stringhe di *b* sull'altro, con la stessa lunghezza

- *traduce* stringhe di *a* in input in stringhe di *b* della stessa lunghezza in output, e viceversa

FST: ε-transizioni

Come gli FSA, anche gli FST possono avere ε-transizioni (o *jump arcs)*

ESEMPIO



- *riconosce* tutte le coppie di stringhe in cui quella sul primo nastro ha tutte *a* e quella sul secondo un numero doppio di *b*

- *produce* le coppie di stringhe …

- *traduce* stringhe di *a* in input in stringhe di *b* di lunghezza doppia

Σ = {a:b,ε:a}

L = {0,abb,aabbbb,aaabbbbbb…}