Linguistica Computazionale

Lezione 2

Strumenti linguistico-formali & informatici

30 Novembre 2007
Cristiano Chesi, chesi@media.unisi.it

Strumenti linguistico-formali & informatici

- **Indice**
  - Grammatiche formali
    - Nozioni di base (grammatiche a struttura sintagmatica, PSG)
    - Gerarchia di Chomsky
    - Descrizioni Strutturali e derivazioni
  - Formalismi applicabili alla MT
    - Regole di Transfer, Interlingua e Ontologie, Grammatiche ad unificazione, Principi e Parametri
  - Fondamenti informatici
    - Macchine di Turing (universali): concetto di computazione e computabilità
    - Dati, programmi, input e output
  - Basi Dati
    - Corpora e database
  - Algoritmi
    - Cicli ed oggetti
    - Ideazione, descrizione, formalizzazione ed implementazione di un algoritmo

Letture, approfondimenti

- **Bibliografia essenziale**
  - Hutchins & Somers (1992) Cap. 2, 3
  - Jurafsky & Martin (2000) Cap. 1, 2
- **Approfondimenti**
  - Turcato D. (1993) *Grammatiche formali e linguaggio naturale*. Calderini Bologna
  - Allegranza, V., Mazzini G. (2000) *Linguistica generativa e grammatiche a unificazione*. Paravia scriptorium.
  - Baker M. (2001) *The Atoms of Language*. Basic Books
  - Lenci, Montemagni & Pirrelli (2005) *Testo e Computer: Elementi di Linguistica Computazionale*. Carocci, Roma
  - Hopcroft, Motwani & Ullman (2001) *Introduction to the automata theory, languages and computation*. Addison-Wesley. Boston
  - Frixione & Palladino (2004) *Funzioni, Macchine, Algoritmi*. Carocci. Roma

Competence = grammatica

- **Competence** (data-structure, natura del problema)
  - di che tipo di struttura dati ha bisogno la conoscenza linguistica?
    - una parola può iniziare per *ma...* (*mare*) ma non per *mr...*
    - la e di case ha un valore diverso da quella di mare
    - "le case sono sulla collina" Vs. *"case le collina sono sulla"
    - il gatto morde il cane > sogg: gatto(agente); verbo: morde(azione); ogg: cane(oggetto)
    - ?il tostapane morde il gatto
    - l'espressione "le case" si riferisce ad un gruppo di case evidente dal contesto (Vs. "delle case")
  - ad ogni livello si devono specificare delle primitive elementari:
    - **fonemi** - tratti segmentali e soprasegmentali
    - **morfemi** - identificazione delle regole combinatorie
    - **parole** - gruppi di morfemi significativi
    - **sintagmi** - gruppi tipizzati di parole che esprimono relazioni
    - **elementi tematici** - paziente, agente...
    - **elementi discorsivi** - convenzioni, relazioni pragmatiche pertinenti…

Livelli di adeguatezza di una grammatica

- **Adeguatezza**: una grammatica deve fornire una descrizione adeguata rispetto alla realtà empirica a cui si riferisce. In particolare si può parlare di adeguatezza a tre livelli:
  - **osservativa**: la lingua definita dalla grammatica coincide con quella che si intende descrivere
  - **descrittiva**: l'analisi grammaticale proposta è in linea con le intuizioni linguistiche dei parlanti fornendo descrizioni strutturali adeguate delle frasi accettabili
  - **esplicativa**: i dispositivi generativi utilizzati soddisfano criteri di plausibilità psicolinguistica e riproducono realmente i meccanismi operanti nell'attività linguistica del parlante. Una grammatica si dice esplicativa quando rende conto anche dell'apprendibilità della lingua.

Come si formalizza una grammatica

(inizio ...)

- **A = Alfabeto**
  insieme finito di caratteri (A* = l'insieme di tutte le stringhe possibili costruite concatenando elementi di A; ε è l'elemento nullo)
- **V = Vocabolario**
  insieme (potenzialmente in)finito di parole, costruite concatenando elementi di A
  ($V \subseteq A^*$)
- **L = Linguaggio**
  insieme (potenzialmente in)finito di frasi, costruite concatenando elementi di V
  ($L \subseteq V^*$)

Come si formalizza una grammatica

(... continua ...)

- Una **grammatica formale** per il linguaggio **L** è un insieme di regole che permettono di **generare/riconoscere** tutte e sole le frasi appartenenti a **L** e(d eventualmente) di assegnare a queste frasi un'adeguata descrizione strutturale.

  Una grammatica formale G deve essere:

  - **esplicita** (il giudizio di grammaticalità deve essere frutto solo dell'applicazione meccanica delle regole scelte)
  - **consistente** (una stessa frase non può risultare allo stesso tempo grammaticale e non grammaticale)

Come si formalizza una grammatica

(... continua ...)

- **Una grammatica formale** G **può essere formalizzata (**grammatica a struttura sintagmatica **o Phrase Structure Grammar, PSG Chomsky 1965), come una quadrupla ordinata** $<V, V_T, \rightarrow, \{S\}>$ **dove:**

  V è il **vocabolario** della lingua

  $\mathbf{V_T}$ è un sottoinsieme di **V** che racchiude tutti e soli gli **elementi terminali** (il complemento di V_T rispetto a V sarà l'insieme di tutti i vocaboli non terminali e sarà definito come $\mathbf{V_N}$)

  $\rightarrow$ è una relazione binaria, asimmetrica e transitiva definita su **V***, detta **relazione di riscrittura**. Ogni coppia ordinata appartenente alla relazione è chiamata **regola di riscrittura**. Per ogni simbolo $A \in V_N \quad \varphi A \psi \rightarrow \varphi \tau \psi$ per qualche $\varphi, \tau, \psi \in V^*$

  {S} è un sottoinsieme di V_N definito come l'insieme degli assiomi che convenzionalmente contiene il solo simbolo **S**.

Come si formalizza una grammatica

(… fine)

- Date due stringhe φ e ψ ∈ V* si dice che esiste una **φ-derivazione di ψ** se φ →* ψ.
- Se esiste una φ-derivazione di ψ allora si può anche dire che **φ domina ψ**. Tale relazione è riflessiva e transitiva.
- Una φ-derivazione di ψ si dice **terminata** se:
  - ψ ∈ V_T*
  - per nessun χ esiste una ψ-derivazione di χ
- Data una grammatica G, una **lingua generata** da G, detta **L(G)**, è l'insieme di tutte le stringhe φ per cui esiste una S-derivazione terminata di φ.

Descrizioni strutturali (cioè alberi sintattici)

- **Una Descrizione Strutturale è una quintupla** <V, I, D, P, A> **tale che:**

  V è un insieme finito dei **vertici** (es. v1, v2, v3…)

  I è l'insieme finito degli **identificatori** (es. S, DP, VP, la, casa…)

  D è la relazione di **dominanza**. È un ordine debole (cioè una relazione binaria, riflessiva, antisimmetrica e transitiva) definita su V

  P è la relazione di **precedenza**. È un ordine stretto (cioè una relazione binaria, irriflessiva, antisimmetrica e transitiva) definita su V

  A è la **funzione di assegnazione**; una funzione non suriettiva da V a I

Capacità generativa e relazioni di equivalenza

- La **capacità generativa** denota l'insieme di oggetti generati dalla grammatica; tale capacità è:
  - **debole** se riferita al solo semplice insieme di frasi generabili
  - **forte** se associa a tali frasi l'appropriata descrizione strutturale
- Due grammatiche si dicono **equivalenti** se sono in grado di generare lo stesso insieme di oggetti.
  Anche qua si può parlare di **equivalenza debole** o **equivalenza forte**

Decidibilità

Un insieme Σ si dice

- **decidibile** (o ricorsivo) se per ogni elemento *e* appartenente all'insieme universo esiste un **procedimento meccanico** *M* che permette di stabilire in un **numero finito di passi** se e ∈ oppure ∉ a Σ (la non appartenenza a Σ determina l'appartenenza al complemento di Σ definito come $\overline{\Sigma}$)
- **ricorsivamente enumerabile** quando esiste un procedura che enumera tutti e soli gli elementi di Σ

Inclusioni tra classi di grammatiche

(inizio ...)

- La **gerarchia di Chomsky** (1956, 59) pone in relazione grammatiche di potenza diversa ponendo restrizioni sulla struttura delle regole:

  Tipo 0: grammatiche non ristrette (**Turing Equivalent**):
  $\alpha \rightarrow \beta$ $(\alpha \neq \varepsilon)$ (es. Augmented Transition Networks)

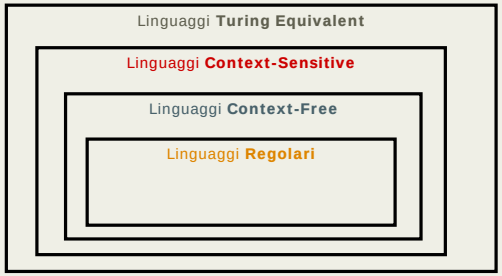
  Tipo 1: grammatiche contestuali (**Context Sensitive**):
  $\alpha A \beta \rightarrow \alpha \gamma \beta$ $(\gamma \neq \varepsilon)$ (es. Tree Adjoining Grammars)

  Tipo 2: grammatiche non-contestuali (**Context Free**):
  $A \rightarrow \gamma$ (es. Phrase Structure Grammars)

  Tipo 3: grammatiche **regolari**:
  $A \rightarrow xB$ (es. Finite State Automata)

Inclusioni tra classi di grammatiche

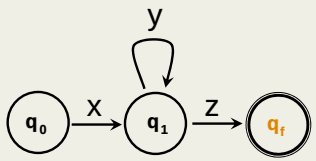
(... fine)



Come si stabilisce l'appartenenza ad una grammatica

- **Pumping lemmas**
  Servono per verificare se una proprietà linguistica può essere catturata da una grammatica oppure no

- **Pumping lemma per le grammatiche regolari**
  $a^n b^n$ non è una stringa generabile da nessuna grammatica regolare (poiché nessuna sottostringa può essere "pompata" indefinitivamente garantendo lo stesso numero di *a* e di *b*)



Dove stanno le lingue naturali?

(inizio ...)

- **Le lingue naturali** non sono generabili da grammatiche regolari **(Chomsky 1956):**

  If A then B (con A e B potenzialmente anch'esse nella forma "if X then Y"... quindi linguaggi di tipo $a^n b^n$)

- **Le lingue naturali** non sono generabili da grammatiche context-free **(Shieber 1985):**

  Jan säit das mer em Hans es huus hälfed aastriiche
  ("famoso" dialetto svizzero tedesco)
  J. dice che noi a H. la casa abbiamo aiutato a dipingere

  Gianni, Luisa e Mario sono rispettivamente
  sposato, divorziata e scapolo
  ("ABC...ABC"... quindi linguaggi di tipo XX)

Dove stanno le lingue naturali?

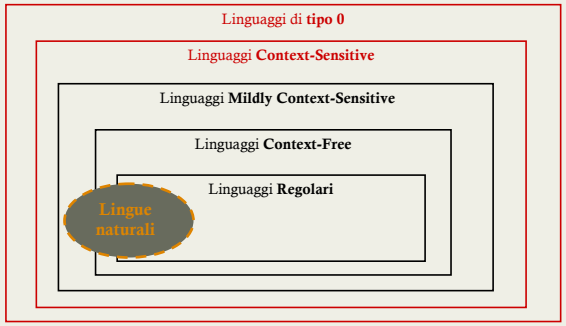
(... continua ...)

Ricorsività nelle lingue naturali, ovvero come fare un uso infinito di mezzi finiti:

- **Incassamento a destra** (ab^n: iterazione):
  [il cane morse [il gatto [che rincorse [il topo [che scappò]]]]]
- **Incassamento centrale** (a^nb^n: counting recursion):
  [il cane [che il gatto [che il topo che scappò], rincorse], morse]
- **Dipendenze cross-seriali** (xx, identity recursion)
  Gianni, Maria e Marco sono rispettivamente sposato, nubile e divorziato

Dove stanno le lingue naturali?

(... fine)

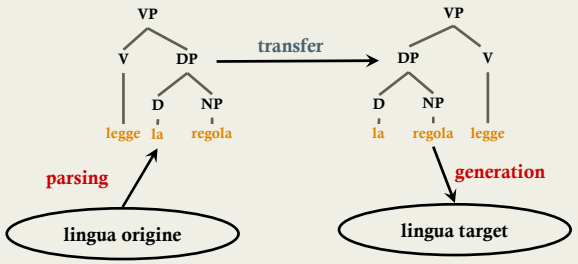


Altri fenomeni linguistici interessanti "catturabili" con CFGs

- **accordo**
  per cogliere fenomeni di accordo si deve ricorrere alla duplicazione delle regole di riscrittura. Ad esempio:
  $D_{pl} \rightarrow i$, $N_{pl} \rightarrow cani$, $D_{sg} \rightarrow il$, $N_{sg} \rightarrow cane$, $DP \rightarrow (D_{sg}\ N_{sg} \mid D_{pl}\ N_{pl})$
- **sottocategorizzazione**
  ci si riferisce allo schema di sottocategorizzazione come alla possibilità di distinguere ulteriormente, all'interno di categorie maggiori (ad esempio la categoria verbale), categorie più precise che rendano conto in modo più adeguato del comportamento dei diversi elementi lessicali: transitivo, intransitivo, inaccusativo o ergativo sono solo tre delle sotto-classi verbali che servono a predire un determinato comportamento verbale (certi approcci, Levin 83, distinguono fino a 183 classi verbali distinte).
  esempi di regole:
  $VP \rightarrow (V_{transitivo}\ DP \mid V_{intransitivo} \mid V_{frasale}\ S \mid V_{modale}\ V_{inf}\ldots)$

Modello del Transfer

- **Conoscenza contrastiva**
  esplicitare le differenze tra le due lingue è il primo passo verso la traduzione.
  Da questo punto di vista occorre una ristrutturazione linguistica per conformarsi alle regole della lingua target

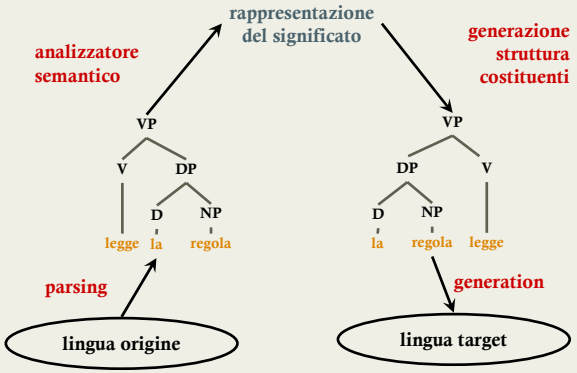


Esempi di Transfer con Context-Free Grammars

- **Inglese** ⇒ **Italiano**
  $DP \rightarrow D_1\ Agg_2\ Nome_3$ ⇒ $DP \rightarrow D_1\ Nome_3\ Agg_2$

- **Giapponese** ⇒ **Inglese**
  $DP \rightarrow relativa_1\ DP_2$ ⇒ $DP \rightarrow DP_2\ relativa_1$

Modello dell'interlingua



Ontologie

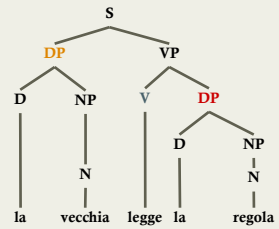
- **Concettualizzazione**
  astrazione (e semplificazione) delle relazioni tra oggetti e concetti in un dominio di conoscenza

- **Ontologia**
  specificazione dettagliata di queste entità e relazioni

  Ogni ontologia definisce un insieme di **classi**, **relazioni**, **funzioni** e **oggetti** costanti all'interno di un dominio discorsivo, esplicitando un'assiomatizzazione in modo da vincolare interpretazioni ed inferenze

Rappresentazione del significato

– **Aggiunta semantica** alle regole Context Free:
  $A \rightarrow \alpha_1\ \alpha_2 \ldots \alpha_n$ $\{f(\alpha_j.sem \ldots \alpha_k.sem)\}$
  vecchia $\{vecchia\}$
  legge $\{\exists e,x,y\ Isa(e, leggere) \wedge legge(e, x) \wedge letta(e, y)\}$



$\exists e$ Isa(e, leggere) ∧ legge(e, la vecchia) ∧ letta(e, la regola)

Grammatiche ad unificazione

(inizio ...)

Formalismi applicabili alla MT

- **Grammatiche basate su restrizioni**
  rappresentazione più efficiente e significativa dell'informazione linguistica
- **formalismi leggermente più "potenti" delle CFG**, con cui render conto in modo
  - **compatto** (quindi più elegante) ed
  - **efficiente**

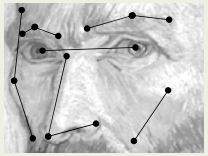
  delle restrizioni linguistiche imposte da fenomeni produttivi quali quelli precedentemente elencati
- **gerarchie di tratti**
  proprietà aggiuntive rispetto alle regole di riscrittura

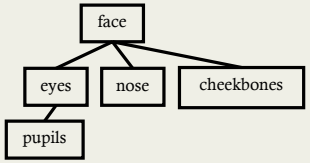
Grammatiche ad unificazione

(... continua ...)

Formalismi applicabili alla MT

- (ma... cosa sono i tratti?)





Grammatiche ad unificazione

(... continua ...)

Formalismi applicabili alla MT

- (ma... cosa sono i tratti?)

a. b. c.* d.* e.*

Il bambino chiude la porta | Il bambino apre la porta | *Bambino il porta chiude la | *I bambino chiudono le porta | *Il bambino chiude

Grammatiche ad unificazione

(... continua ...)

Formalismi applicabili alla MT

- **Struttura di tratti (FS, Feature Structures)**
  è un insieme di coppie del tipo **tratto > valore**
  (es. numero > singolare)
- due tipologie di formalizzazione (equivalenti) delle coppie **tratto > valore**:

Matrice di Attributi e Valori (AVM, Attribute-Value Matrix)

$$\begin{bmatrix} \text{Num} = \text{Sing} \\ \text{Gen} = \text{Femm} \\ \dots \\ \text{Tratto}_n = \text{Valore}_n \end{bmatrix}$$

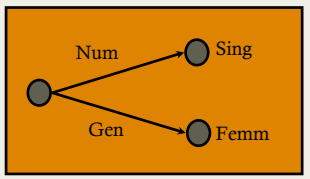


diagramma ad archi orientati ed etichettati (DAG, Direct Acyclic Graph)

Grammatiche ad unificazione

(... continua ...)

Alcune proprietà interessanti delle strutture di tratti:

- **parzialità**, **maggiore** o **minore specificità**, ovvero alcuni elementi posso restare non specificati, ad esempio il genere:

$$N\begin{bmatrix}\text{Num} = \text{Sing}\\ \text{Gen} = [\,]\end{bmatrix}$$

- la struttura delle AVM può essere **rientrante**, cioè un tratto che ha una qualche significatività dal punto di vista empirico, può essere definito da più sottotratti, come nel caso dell'accordo:

$$\begin{bmatrix}\text{Cat} = \text{N}\\ \text{Accordo}\begin{bmatrix}\text{Num} = \text{Sing}\\ \text{Gen} = \text{Femm}\end{bmatrix}\end{bmatrix}$$

Grammatiche ad unificazione

(... continua ...)

Altre proprietà interessanti delle strutture di tratti:

- **percorsi**, il valore di un tratto è definito in base ad un percorso univoco che lo identifica, cioè una lista di tratti lungo la struttura del tipo: accordo>num>sing
- **condivisione di tipo** (**type sharing**), una struttura può essere condivisa tra più elementi anche se i valori non lo sono
- **condivisione di occorrenza** (**token sharing**), l'occorrenza di un determinato valore può essere condivisa in tal caso può essere indicato con l'uso di un indice, es [1]:

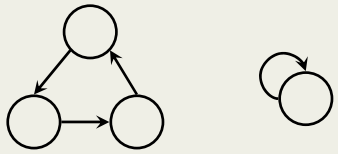
$$\begin{bmatrix}\text{Cat} = \text{S}\\ \text{testa}\begin{bmatrix}\text{Accordo}[1]\begin{bmatrix}\text{Num} = \text{Sing}\\ \text{Per} = 3\end{bmatrix}\\ \text{Soggetto}\left[\text{Accordo}[1]\right]\end{bmatrix}\end{bmatrix}$$

Grammatiche ad unificazione

(... continua ...)

Alcune proprietà interessanti delle strutture di tratti:

- **significatività empirica**, i tratti sembrano catturare adeguatamente, almeno a livello descrittivo, fenomeni linguisticamente produttivi
- **aciclicità**, i grafi non possono essere ricorsivi



Grammatiche ad unificazione

(... fine)

Sussunzione

stabilisce una relazione ordinata tra due strutture di tratti FS; la FS più **generica** sussume quella **specifica**. Si può quindi dire che:
$FS_a \sqsubseteq FS_b$

sse FS_b ha tutti i tratti di FS_a nella stessa configurazione strutturale e con uguali assegnazioni di valore

Unificazione

permette di combinare le informazioni per rappresentarle in formato più compatto e significativo:
$FS_a \sqcup FS_b = FS_x$ (se esiste) tale che FS_x è la più generale delle FS sussunte da FS_a e FS_b

Da regole a principi e parametri

(inizio ...)

- **regole** — **principi & parametri**
  specifiche e valide — universali linguistici +
  per una sola lingua — settaggio parametri di variazione
- Ricerca di una migliore **adeguatezza esplicativa** oltre che **descrittiva**
- **Obiettivo**: cogliere gli universali linguistici descrivendo precisamente la limitata variabilità sintattica
- I **principle-based parsers** (Barton 1984, Berwick e Fong 90, Stabler 92) si ispirano a queste idee:
  - i principi della grammatica sono **assiomi** per il parser
  - il parser è un **sistema deduttivo** che inferisce le espressioni grammaticali e la loro struttura partendo da tali assiomi

Da regole a principi e parametri

(... fine)

- **regole**

  regola passivo → frase passiva
  regola dativo → frase dativa
  regola di focalizzazione → frase focalizzata
  ...

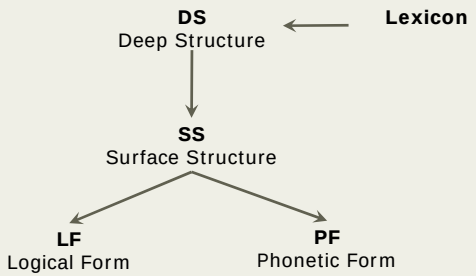
- **principi & parametri**

- potenzialmente una decina di principi + pochi parametri possono generare migliaia di regole

Principi e parametri

(inizio ...)

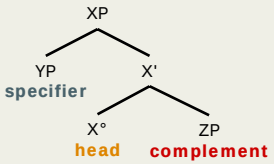
– Modello a "T"



Principi e parametri

(... continua ...)

- Alcuni principi
  - **X' theory**



  - **θ - criterion**
    ogni argomento deve ricevere uno ed un solo ruolo tematico (e ogni ruolo tematico è assegnato ad uno ed un solo argomento)
  - **Case filter**
    ogni NP lessicale deve ricevere un caso (P e V_{finito} assegnano caso)

Principi e parametri

(... continua ...)

- Altri principi
  - **Move α**
    una categoria può muoversi in qualsiasi momento, ovunque
  - **Free indexation**
    indici sono liberamente assegnati alle categorie in posizione A(rgomentale)
  - **Binding theory**
    condizione A – Un'anafora (es. *se stessa*) è legata nel suo dominio di legamento (binding domain)
    condizione B – Un pronome (es. *lei*) è libero nel suo dominio di legamento
    condizione C – Un'espressione referenziale (es. *Maria*) è sempre libera

Principi e parametri

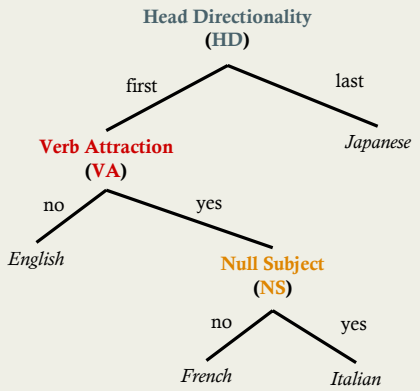
(... fine)

- **Generatori**
  principi che producono più strutture di quante non ne ricevano in input:
  - **Move α**
  - **Free indexation**
  - **…**
- **Filtri**
  principi che selezionano solo parte delle strutture che ricevono in input:
  - **X' theory**
  - ***θ* - criterion**
  - **Case filter**

Principi e parametri

(... fine)

- Parametri (Baker 2002)



Macchine di Turing (universali)

- **Concetto di computabilità**

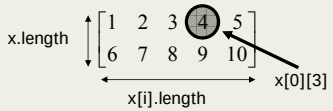
  (informalmente) per **computazione** si intende la specificazione della **relazione tra un input ed un output**. Questa relazione può essere definita a vari livelli, ma fondamentalmente consiste nella descrizione di una **serie di stadi** intermedi in cui l'informazione di input può venire trasformata prima di raggiungere la forma dell'output e nella definizione delle specifiche di trasformazione. Un problema computazionale tenta quindi di ridurre ogni input ad output seguendo una serie di passi consentiti dal modello computazionale.

- **La tesi di Turing-Church**

  ogni compito computazionale che può essere realizzato da un qualsiasi dispositivo fisico può essere realizzato anche da una macchina di Turing. Inoltre se il dispositivo fisico riesce a completare il compito in F(n) passi, con *n* uguale alla dimensione dell'input, la macchina di Turing ci riuscirà in G(n) passi, con F che differisce da G di al massimo un polinomiale.

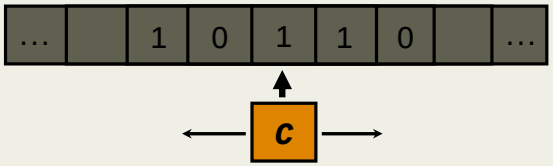
Input e Output

- Affinché un programma possa accettare un **input**, questo input deve essere conforme a certe specifiche dichiarate a priori:
  - **formato** (codifica, es. ASCII, UTF-8, binary)
  - **struttura** (impacchettamento, es. header, TCP-IP... DTD)
- Un **output** deve o sottostare agli stessi standard oppure essere human readable (di solito si preferisce avere output in formato standard, es. XML e poi gestire la visualizzazione, cioè l'interfaccia, con moduli specifici).
- Una funzione (es. f(x) = y) è definita nel suo dominio di applicazione (input: x ∈ X) e di proiezione (output (range) : y ∈ Y)
  - String x = "scuola"
  - Int x = 14
  - Int[][] x = {{1, 2, 3, 4, 5 }, {6, 7, 8, 9, 10}}



Macchine di Turing

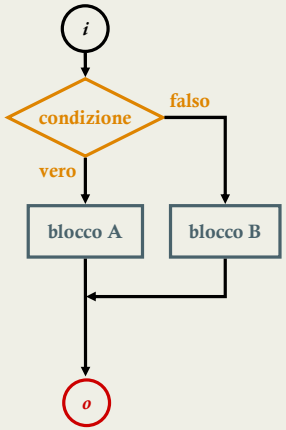
- nastro infinito diviso in celle
- alfabeto ***A*** (di almeno 2 elementi, ad esempio ***A*** = {0, 1})



- cursore ***C*** (che scorre a sinistra e a destra, che può leggere, cancellare e scrivere un carattere)
- insieme finito ***Q*** di stati (q_0, $q_1 \ldots q_n$)
- input finito ***I*** costituito da caratteri in ***A***
- insieme finito ***S*** di stati della macchina descritte da quintuple del tipo $<q_i abv q_j>$ dove $q_i, q_j \in Q$; $a,b \in A$; v = {destra, sinistra}

Flow charts

- Grafo orientato etichettato **costituito da:**
  - un **ingresso** (dove viene introdotto l'input *i*)
  - una o più **uscite** (da cui viene, eventualmente, recuperato un output *o*)
  - un insieme finito di **blocchi di istruzioni** tali che ogni istruzione è del tipo X = Y, X = X+1, X = X-1
  - un insieme finito (possibilmente nullo) di blocchi speciali, detti **condizioni** tali da presentare interrogazioni booleane del tipo X = Y?
  - un insieme finito di **connettori** che collegano i blocchi, tali che da ogni blocco esce 1 ed 1 sola freccia, mentre dal blocco condizionale escano 2 frecce



Modularità

(inizio ...)

- **Macchine di Turing** e **flow charts** sono equivalenti: esprimono la stessa classe di funzioni: le funzioni computabili.
- Entrambi i formalismi godono della proprietà della **composizionalità** ($M_1 \bullet M_2$). Questo ci garantisce che un algoritmo può in realtà essere il risultato di una composizione di più macchine di Turing (o di più flow charts).
- Si dice "**divide et impera**" il paradigma di programmazione che suggerisce di scomporre un problema nelle sue sottoparti prima di iniziare a pensare agli algoritmi che lo risolvono.

Modularità

(… fine)

Fondamenti informatici

- Una prima distinzione può essere fatta tra **dati** e **programmi** (**decoupling**) dove per **dato** si intende un oggetto "inerte" che può essere oggetto (input) o risultato (output) della computazione. La parte più strettamente algoritmica viene descritta dal **programma**.
- In genere modularizzare un problema conviene per i seguenti motivi:
  - **divisione del lavoro** (capire cosa può essere svolto in parallelo)
  - **riutilizzo** (capire quali saranno le funzioni più generali e riutilizzarle)
  - **analisi modulare** (cicli di sviluppo più rapidi: progetto > implementazione > testing > debugging …)
  - **modifiche locali** (maggiore scalabilità)

La struttura dei dati

Fondamenti informatici

- **Linguaggi naturali e linguaggi "taggati"**
  - parentesi [[A] [B C]]
  - HTML 123 Mario Rossi
  - XML <studente> <id> 123 </id>
    <nome> Mario Rossi </nome>
    </studente>

Cosa sono, che struttura hanno

Basi dati

- collezioni **finite** di informazioni, **omogenee** e **rappresentative** rispetto ad un dominio, raccolte in un modo **sistematico**, in **condizioni controllate** in modo da riflettere la **reale distribuzione** (quantitativa e qualitativa) dei fenomeni linguistici che si intendono studiare
  - **non strutturate** (unica informazione presente è l'informazione linguistica del testo)
    es. files di testo con formattazione non significativa (colonne, giustificazione…)
  - **strutturate** (convenzioni precise indicano la natura dei dati linguistici)
    es. database, testo taggato
  - **semistrutturate** (convenzioni implicite forniscono implicitamente informazioni (extra)linguistiche)
    es. pagine html, testi formattati (titoli, paragrafi, sottoparagrafi, grassetti, corsivi ecc.)

Corpora linguistici

(inizio …)

Basi dati

- **Brown corpus** (Francis and Kucera, 1964)
  - circa un milione di parole rappresentativo dell'inglese americano scritto (500 testi del 1961)
  - i testi sono raccolti in 15 categorie:
    - A. stampa: reportage (44 texts)
    - B. stampa: editoriali (27 texts)
    - C. stampa: periodici (17 texts)
    - D. religione (17 texts)
    - E. arti e mestieri (36 texts)
    - F. tradizioni popolari (48 texts)
    - …
  - Esempio:
    A01 0010 The Fulton County Grand Jury said Friday an investigation
    A01 0020 of Atlanta's recent primary election produced "no evidence" that
    A01 0030 any irregularities took place. The jury further said in term-end
    A01 0040 presentments that the City Executive Committee, which had over-all
    A01 0050 charge of the election, "deserves the praise and thanks of the
    A01 0060 City of Atlanta" for the manner in which the election was conducted.

Esempio di corpus linguisticamente strutturato

(... continua ...)

– **Penn Treebank** (Marcus & al. , 1989-1992)

- **1 milione di parole** provenienti da articoli del 1989 del Wall Street Journal
- Un campione di materiale proveniente da ATIS-3 (Automatic Terminal Information Service)
- Etichettatura secondo lo "**standard**" Treebank II style
- esempio:
  (S (PP (IN **Of**) (NP (NN **course**))) (, ,) (S (S (NP (DT **some**) (PP (IN **of**) (NP (PRP$ **my**) (NN **color**) (NNS **values**)))) (AUX (VBP **do**)) (NEG (RB **not**)) (VP (VB **match**) (NP (NP (DT **the**) (JJ **old**) (NN **Master**)) (POS **'s**)))) (CC **and**) (S (NP (DT **the**) (NN **perspective**)) (VP (VBZ **is**) (ADJP (JJ **faulty**)))) (CC **but**) (S (NP (PRP **I**)) (VP (VBP **believe**) (S (NP (PRP **it**)) (AUX (TO **to**)) (VP (VB **be**) (NP (DT **a**) (JJ **fair**) (NN **copy**)))))))

Esempio di corpus linguisticamente semi-strutturato

(... fine)

- **Childes** (MacWhinney & Snow, 1985)
  - (**Chi**ld **L**anguage **D**ata **E**xchange **S**ystem) è un archivio di trascrizioni spontanee di bambini (solitamente dai 14 mesi ai quattro anni di età) che interagiscono con adulti in varie situazioni. Generalmente ogni trascrizione si riferisce ad una conversazione di durata variabile dai 20 ai 60 minuti.
  - Le trascrizioni sono codificate secondo il formato standardizzato, detto **CHAT**

```
@UTF8
@Begin
@Participants:   CHI Cam Target_Child, DON Mother
@Age of CHI:     3;4.9
@Sex of CHI:     female
@Birth of CHI:   3-MAY-1988
@Date:           12-SEP-1991
*DON:            quale volevi ?
*CHI:            io volevo questo .
*DON:            si ma cosa, che canzoni ci sono, sopra .
*CHI:            non lo so .
*DON:            come non lo sai ?
                 […]
@End
```


A cosa servono i corpora

– Informazioni quantitativamente significative per implementare **sistemi esperti** o per l'**estrazione di grammatiche**
  - registrazioni telefoniche (call center)
  - corpora taggati
  - ...

– **Analisi linguistiche** specifiche
  - acquisizione prima lingua
  - acquisizione seconda lingua
  - soggetti con disturbi linguistici (Specific Language Impairment, sordi, afasici ...)
  - ...

Database relazionali

– **tabelle**

studenti

ID	nome	cognome	mail	...
1	**aldo**	**rossi**	**aldo@…**	
2	**giovanni**	**bianchi**	**gio@…**	
3	**giacomo**	**verdi**	**gia@…**	
...				

– **relazioni (o link relazionali o join)**

studenti

ID	nome	cognome	stato
1	**aldo**	**rossi**	**1**
2	**giovanni**	**bianchi**	**2**

iscrizione

ID	descrizione
1	**presente**
2	**assente**

Database relazionali - Interrogazione

- **Structured Query Language (SQL)**
  Linguaggio di interrogazione e manipolazione di database
  - Data Manipulation Language (DML)
    - **select**
      select * from studenti where id > 1
    - **insert**
      insert into studenti (nome, cognome) values (“mario”, “rossi”)
    - **delete**
      delete from studenti where id=10
    - **update**
      update studenti set nome=“gianni” where id=11
  - Data Definition Language (DDL)
    - **create database**
    - **create table**
    - **drop database**

Interrogazioni su corpus testuali

- **Espressioni Regolari**
  - notazione algebrica per definire insiemi di stringhe di testo (linguaggi di **tipo 3**).
  - Il cuore dell’espressione regolare è il **pattern di identificazione** composto da caratteri alfanumerici (compresi segni di spaziatura e di interpunzione) e da segni speciali volti a stabilire le relazioni tra i caratteri del pattern.

Espressione Regolare	Corrispondenza	Es. pattern identificato
[Dd]uomo	Duomo oppure duomo	Il duomo è nella piazza
[^a-z]	tutto fuorché lettere minuscole	Il duomo è …
sali?ta	salita oppure salta	Marco deve saltare
sal.ta	accetta ogni carattere tra le i e la t	Marco saluta
bu*	b seguito da un numero imprecisato (anche nullo) di u	buuuuuu! oppure b!
^L Vs. a$	^ = inizio stringa; $ = fine stringa	La casa
cas(a|e)	| è equivalente alla disgiunzione logica	Marco vive in un casale
*	il backslash è il simbolo di escape	A*

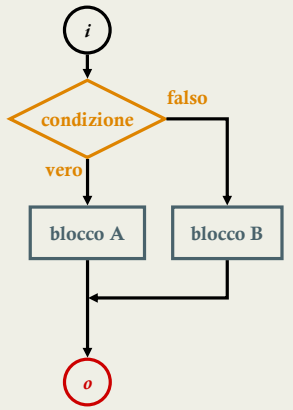
Note su Eliza

- **Espressioni Regolari e l’operazione di Sostituzione**
  - La **sostituzione** è un’operazione che permette di sostituire l’occorrenza di un’espressione regolare con un’altra espressione regolare e può essere definita come segue:
  - **s/espressione_regolare1/espressione_regolare2/**
  - **s/ www\.[a-z]*\.com / www\.pe{2}\.com/**
- **Registri**: se si usano più blocchi di operatori (ogni parentesi tonda delimita un blocco), si può riutilizzare l’espressione trovata da un determinato blocco nell’espressione da sostituire, facendo riferimento all’ordine dei blocchi nel pattern di ricerca:
  - **s/ la (casa|macchina) è stata comprata da (Maria|Gianni)/ \2 ha comprato la \1 /**
    permette di costruire la forma attiva (Gianni ha comprato la casa) della frase passiva (la casa è stata comprata da Gianni).
- operazioni di sostituzione in ELIZA:
  - s/ sono [.* |](depress[o|a]|triste)/sono spiacente di sapere che sei \1/
  - s/ sono tutt[i|e] (.*) /in che senso sono \1?/
  - s/ sempre / potresti far riferimento ad un esempio specifico?

Controllo di flusso condizionale: If … else …


```
if (condizione-booleana) {
  blocco A
}
else {
  blocco B
}
```

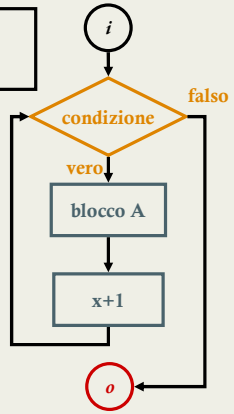
```
if (i < 10) {
  i + 1;
  o = i;
}
else {
  o = 10;
}
```



Cicli determinati

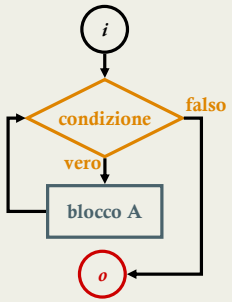

```
for (input; condizione-booleana; incremento) {
  blocco
}
```

```
for (int i = 0; i < 10; i++) {
  System.out.println(i);
}
```



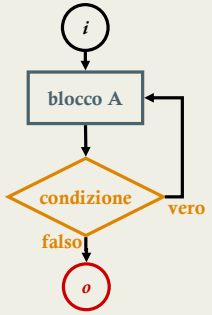
Cicli indeterminati

- Cicli **While**




```
while (boolean-cond) {
  blocco
}
```

- Cicli **Do … While**




```
do {
  blocco
} while (boolean-cond)
```


Interruzioni dei cicli

- **Break** interrompe l'esecuzione di un ciclo ed esce definitivamente dal ciclo (while, do while, for)

- **Continue** interrompe la corrente iterazione e ritorna all'inizio del ciclo, iniziandone un'altra.

```
for (int i = 0, i < 20; i++) {
  if (i = 0)
    continue;
  if (i % 2 == 0)
    break;
  System.out.println(i);
}
```


Strutture dati ed OGGETTI

- **Oggetto**
  entità costituita da **proprietà** (il **valore** assegnato a tale proprietà in un determinato istante di tempo determina lo **stato** dell'oggetto) e **comportamenti** (metodi o procedure di modificazione dei dati proprie dell'oggetto).
  Un oggetto complesso è un oggetto costituito da altri oggetti (funzione di **estensione**).

- **Identità**
  si dice **Object IDentifier (OID)** l'identificativo unico dell'oggetto, indipendente dai valori che tale oggetto assume (avere lo stesso identificativo è diverso dall'avere gli stessi valori!)

Strutture dati ed OGGETTI – EREDITARIETA'

- Gli oggetti possono essere definiti **gerarchicamente**: ogni oggetto gerarchicamente inferiore eredita proprietà e metodi dagli oggetti padri
- Nuovi metodi/proprietà possono essere inclusi nei figli, gli stessi metodi ereditati possono essere ridefiniti (**overriding**)
- si può parlare di ereditarietà semplice (classi>sottoclassi) oppure ereditarietà multipla (reticolo aciclico)

Dati e programmi

- Difficile progettare sistemi **top-down** (prime suddivisioni funzionali troppo "lontane" dalla reale implementazione del sistema: difficile prevedere modularizzazioni efficienti)
- (di solito) è meglio **partire dai dati** e organizzare i moduli allo stesso livello di astrazione, cercando di individuare le dipendenze ed eventualmente raffinando parallelamente tutte le strutture.

Concetti fondamentali della lezione di oggi

- **Cos'è una Grammatica Formale e cosa serve per specificarne una**
  - Regole di riscrittura e ricorsività
  - Restrizioni sulle regole di riscrittura per creare classi di grammatiche generativamente più o meno potenti (gerarchia di Chomsky)
  - dove stanno i linguaggi naturali e quali sono i limiti di decidibilità
- **Espressione dei tratti e le Grammatiche ad Unificazione**
- **Teoria della variazione linguistica: Principi e Parametri**
- **Idea di Computazione e macchine di Turing**
  - **diagrammi di flusso e programmazione a oggetti**
- **Corpora e database**
  - **come si interrogano (SQL ed espressioni regolari)**

Prossima lezione

(Lunedì 3 Dicembre, ore 16-19, Aula 456, Palazzo S.Niccolò)

- Analisi Morfologica
  - Lassico generativo
  - Morfologia a due livelli
  - Analisi morfologica con FSA
  - Stemming
- Normalizzazione dell'input
  - Classificazione errori
  - Correzione ortografica
  - T9