Domande di LFT – Parte I

Slide 1

- **Cos'è un alfabeto?**

E' un insieme finito e non vuoto di elementi detti simboli o
caratteri con cui comporre stringhe, per indicarlo si usa convenzionalmente il simbolo 'sommatoria'. (ES. alfabeto binario,insieme di tutte le lettere minuscole).
La cardinalità di un alfabeto e' il numero di simboli dell'alfabeto.

- **Cos'è una stringa?**

E' una sequenza finita di simboli scelti da un alfabeto che rispetta le regole di costruzione di una certe grammatica. Stringa vuota: priva di simboli,indicata con 'ε', fa parte di ogni alfabeto.

- **Quando due stringhe sono uguali?**
  Se i loro caratteri, letti ordinatamente da sx a dx, coincidono.
- **Cos'è una sotto stringa?**
  La stringa Y è una sotto stringa della stringa X se esistono delle stringhe U e V tali che X = UYV
- **Cos'è un prefisso?**
  La stringa Y è un prefisso della stringa X se esiste una stringa V tale che X = YV
- **Cos'è un suffisso?**

La stringa Y è un suffisso della stringa X se esiste una stringa U tale che X = UY, un suffisso e' una sotto stringa in cui v = ε

- **Stringa propria:** una sotto stringa (prefisso, suffisso) di una stringa e' propria se non coincide con la stringa vuota o con la stringa stessa.
- **Cos'è la concatenazione? Quali proprietà ha?**

La concatenazione di due stringhe è la stringa formata dai simboli della prima stringa seguiti da quelli della seconda.
Non è commutativa, e' associativa,ha un elemento neutro ovvero ε, la lunghezza della stringa concatenazione è la somma delle lunghezze delle due stringhe.

- **Cos'è la riflessione? Quali proprietà ha?**

La riflessione è la stringa ottenuta scrivendo i caratteri in ordine inverso.
Il riflesso del riflesso di una stringa è la stringa stessa.
La riflessione della concatenazione di due stringhe è la concatenazione inversa delle loro riflessioni. ((xy)^r = y^rx^r)
La riflessione della stringa vuota è la stringa vuota
La riflessione ha la precedenza sul concatenamento.

- **Cos'è la potenza?**

La potenza n-esima della stringa X è il concatenamento di X con se stessa N volte. (ES.(abbc)^3 = abbcabbcabbc
La potenza ha precedenza sul concatenamento.

- **Cos'è un linguaggio?**

Un linguaggio è un insieme di stringhe di un alfabeto. La cardinalità' di un linguaggio e' il numero delle sue stringhe. Un linguaggio e' finito se la sua cardinalità' e' finita (vocabolario). Il linguaggio vuoto (Φ) non contiene alcuna stringa.
L2 dei linguaggi L1 ed L2 e' l'insieme delle stringhe che
{ab, aa, cb} = {ab, abc, aa, cb})
L'intersezione L1 ∩ L2 dei linguaggi L1 ed L2 e' l'insieme delle stringhe
che appartengono sia a L1 che a L2 ({ab, abc} ∩ {ab, aa, cb} = {ab})
La differenza L1 − L2 del linguaggio L1 meno il linguaggio L2 e'
l'insieme delle stringhe di L1 che non appartengono a L2({ab, abc} − {ab, aa, cb} = {abc})
L2) se
tutte le stringhe appartenenti a L1 appartengono anche a L2
Il linguaggio L1 e' propriamente incluso nel linguaggio L2 (notazione
L2) se tutte le stringhe di L1 appartengono ad L2 ed almeno una
stringa di L2 non appartiene ad L1
Due linguaggi sono uguali se contengono lo stesso insieme di stringhe

- **Quando un linguaggio è privo di prefissi?**

Se non contiene i prefissi delle sue stringhe.

L ∩ Prefissi (L) = **Φ**

- **Cosa sono la riflessione, la concatenazione e la potenza di un linguaggio?**

La riflessione di un linguaggio L è l'insieme delle stringhe riflesse di L({ab, abc} R = {ba, cba})
La concatenazione di due linguaggi L1 e L2 è l'insieme ottenuto concatenando in tutti i modi possibili le stringhe di L1 con le stringhe di L2({ab, abc}{ab, aa, cb} = {abab, abaa, abcb, abcab, abcaa, abccb})
La potenza n-esima di un linguaggio L è il concatenamento di L con se stesso N volte. ({ab, abc}^2 = {abab, ababc, abcab, abcabc})

- **Cos'è la chiusura di Kleene di un linguaggio L? Quali proprietà ha?**

E' l'unione di tutte le potenze di L. (L^*)
L*), idempotenza((L*)* = L*), commutatività della riflessione con la chiusura di Kleene((L*)^R =(L^R)*) e con la potenza((L^m)^R = (L>L)^R = L^R >L^R = (L^R) ^m)

- **Che cos'è la chiusura non riflessiva?**

Rispetto al concatenamento del linguaggio L(L^+) e' l'unione di tutte le potenze positive di L

- **Che cos'è il linguaggio universale su un alfabeto Σ ?**
  E' la sua chiusura di Kleene. {a,b}* = { ε , a, b, aa, bb, ab, ba,. . .}
- **Che cos'è il complemento di un linguaggio L su un alfabeto Σ rispetto ad un alfabeto Δ?**
  E' la differenza tra **Δ*** ed L (¬{ab,ba} {a, b} = {ε, a, b, aa, bb, aaa, . . .})

Slide 2

- **Che cos'è un espressione regolare ?**

Un'espressione regolare r su un alfabeto **Σ** è una stringa costruita sull'alfabeto
{ ., | , *, Φ}

- **Cos'è un linguaggio regolare?**

E' un linguaggio per il quale esiste un'espressione regolare che lo denota.

- **Il pumping lemma:** se un linguaggio è regolare, allora ogni stringa sufficientemente lunga nel linguaggio ha una sottostringa non vuota che può essere ''replicata", ossia ripetuta per un qualunque numero di volte senza che le stringhe risultanti escano dal linguaggio. Si può dunque usare il pumping lemma per dimostrare che diversi linguaggi non sono regolari.
- **Operazioni che preservano la regolarità:** molte operazioni, applicate ai linguaggi regolari, producono come risultato un linguaggio regolare. Tra queste ci sono l'unione, la concatenazione, la chiusura, l'intersezione, la complementazione, la differenza, l'inversione, l'omomorfismo (sostituzione di ogni simbolo con una stringa associata) e l'omomorfismo inverso.

Slide 3

- **Cos'è una grammatica context free?**

Una CFG è un modo di descrivere un linguaggio per mezzo di regole ricorsive chiamate produzioni. Una CFG consiste da 4 componenti: CFG=(V,T,P,S), dove V= insieme delle variabili, T= insieme dei terminali, P = insieme delle produzioni,S= simbolo iniziale. Ogni produzione consiste di una variabile di testa c di un corpo, formato da una stringa di zero o più variabili e terminali.

- **Cos'è un albero sintattico?**

è un albero che mostra gli elementi essenziali
di una derivazione. I nodi interni sono etichettati da variabili e le foglie sono etichettate da terminali o da e. Per ogni nodo interno deve esistere una produzione tale
che la testa della produzione è l'etichetta del nodo, e le etichette dei suoi nodi figli,
lette da sinistra a destra, formano il corpo della produzione.

- **Grammatiche ambigue:** per determinate CFG è possibile trovare una stringa terminale con più di un albero sintattico o, il che è equivalente, più di una derivazione

a sinistra o più di una derivazione a destra. Una grammatica di questo tipo è detta ambigua.

- **Eliminazione dell'ambiguità:** per molte grammatiche utili, è possibile trovare una grammatica non ambigua chc genera lo stesso linguaggio. Purtroppo la grammatica non ambigua è spesso più complessa della più semplice grammatica ambigua per il linguaggio. Esistono anche alcuni linguaggi liberi dal contesto, di solito piuttosto artificiali, che sono

inerentemente ambigui, vale a dire che ammettono solo grammatiche ambigue.

- **Parser:** il concetto di grammatica libera dal contesto è essenziale per realizzare compilatori e altri strumenti per linguaggi di programmazione. Strumenti come YACC prendono una CFG come input e producono un parser, il componente di un compilatore chc estrae la struttura del programma in compilazione.
- **Quando due grammatiche sono debolmente equivalenti?**

Se generano lo stesso linguaggio

- **Terminali / variabili non definite / non raggiungibili.**

Terminali: simboli della grammatica che non possono essere sostituiti con le loro produzioni
Variabile non definita: Non terminale che compare nella parte destra di una produzione senza la presenza della relativa regola nella grammatica
Variabile non raggiungibile: Se non è possibile "arrivare" alle sue produzioni partendo da qualsiasi altro non terminale nella grammatica.

- **Qual è l'algoritmo per ottenere una grammatica pulita?**

1) Eliminare non-terminali NON definiti
2) Eliminare non-terminali NON RAGGIUNGIBILI
3) Eliminare **ε-**produzioni
4) Eliminare produzioni unitarie (regole di riscrittura)

- **Quando una produzione è detta unitaria? Come si elimina?**

Quando ha, nella parte destra, solo un altro non-terminale. [es. A-> D]
Si elimina sostituendo il non terminale nella parte destra con le sue produzioni.

- **Quando una grammatica è in forma normale non annullabile?**

Quando non ha terminali non definiti e non ha terminali annullabili [è ammessa grammatica con starter contenente **ε** se il linguaggio ammette la stringa vuota**]**

- **Quando una grammatica è pulita?**

Quando non ha non-terminali "inutili" e non ammette derivazioni circolari

- **Quando una grammatica è in forma normale di Chomsky?**

Quando le sue produzioni sono nella forma di :
regole omogenee binarie es. A-> BC
regole terminali unitarie es A->a.
Quando la regola S->**ε** è accettata solo se il linguaggio contiene la stringa vuota.
Se vale la regola superiore, S non compare nelle parti destre delle produzioni.

- **Quando una grammatica è in forma normale di Greybach?**
  Se la parte destra di tutte le produzioni inizia con un simbolo terminale, eventualmente seguito da alcune variabili. Unica eccezione ammessa è la presenza della stringa vuota (epsilon, ε) come appartenente al linguaggio descritto.

Slide 4

- **Quando una derivazione è ricorsiva?**

Quando un non terminale produce in almeno un passo una stringa che contiene il non terminale stesso.

- **Cos'è una derivazione leftmost / rightmost?**

E' *LEFTMOST* [canonica sx] se ad ogni passo viene sostituito il non terminale più a SX
E' RIGHTMOST [canonica dx] se ad ogni passo viene sostituito il non terminale più a DX

- **Quando una frase è ambigua?**

Si può dire che una frase è ambigua se ha più di una derivazione canonica [sx o dx]

- **Tipi di ambiguità.**

Ricorsione Bilaterale ($E \rightarrow E + E \mid i$)
Ricorsione a sx e a dx ($A \rightarrow aA \mid Ab \mid c$)

- **Quando un linguaggio è inerentemente ambiguo?**

Se tutte le grammatiche che lo generano sono ambigue

- **Quando due grammatiche sono fortemente equivalenti?**

Quando generano lo stesso linguaggio [quindi sono debolmente equivalenti] e assegnano ad ogni frase alberi sintattici isomorfi.

- **L'equivalenza forte / debole è decidibile?**

L'equivalenza DEBOLE è INDECIDIBILE
L'equivalenza FORTE è DECIDIBILE

Slide 5

- **Quando una grammatica è lineare?**

Quando è una grammatica libera dal contesto (*non contestuale*) la cui la parte destra delle produzioni contiene al massimo un non terminale**.**

- **Quando una grammatica è unilineare?**

Quando il non-terminale è "estremo" della parte destra della produzione, quindi prima o dopo il non-terminale non ci sono altri simboli.

- **Quando una grammatica è strettamente unilineare?**

Quando, riferendosi al non terminale "estremo" proprio delle grammatiche unilineari, prima o dopo lo stesso c'è al massimo un solo simbolo

- **La classificazione di Chomsky.**

La classificazione o gerarchia di Chomsky è una categorizzazione che divide le grammatiche in base alla loro capacità generativa, in quattro classi.

TIPO 0: Ricorsivamente Enumerabili [Macchine di Turing]
TIPO 1: Contestuali [Automi Limitati Lineari]
TIPO 2: Liberi [Automi a pila]
TIPO 3: Regolari [Automi a stati Finiti]

Slide 6

- **Proprietà di chiusura delle grammatiche libere.**
  Unione, concatenamento e chiusura di Kleene.
- **Proprietà del pompaggio per i linguaggi liberi.**

Riguarda la presenza di successioni di stringhe che presentano regolarità ben definite. In particolare ogni stringa sufficientemente lunga di un tale linguaggio contiene una o più porzioni che possono essere ripetute un numero arbitrario di volte ottenendo ancora una stringa appartenente al linguaggio.

- **Cos'è una derivazione autoinclusiva?**

Una produzione del tipo:
$A \rightarrow^* uAv$ con $u \neq \varepsilon$ e $v \neq \varepsilon$.

- **Distinguibilità di stati:** due stati di un DFA sono distinguibili se esiste una stringa di input che porta solo uno dei due in uno stato accettante. Partendo soltanto dal fatto che le coppie formate da uno stato accettante e uno non accettante sono distinguibili, e aggiungendo come nuove coppie quelle i cui successori su un certo simbolo di input sono distinguibili, possiamo scoprire tulle le coppie di stati distinguibili.
- **Automi a stati finiti deterministici (DFA):** E' una quintupla $< Q, \Sigma, \delta, q0, F >$, un DFA ha un insieme finito di stati (Q)e un insieme e finito di simboli di input(Σ). Uno degli stati è iniziale(q0); zero o più stati sono accettanti(F). Una funzione di transizione (δ) stabilisce come cambia lo stato a fronte di un simbolo di input.
- **Automi a stati finiti non deterministici (NFA):** gli NFA si distinguono dai DFA perché possono avere un numero arbitrario, anche zero, di transizioni con la stessa etichetta uscenti dallo stesso stato.
- **Minimizzazione di automi a stati finiti deterministici:** si possono ripartire gli stati di un DFA in blocchi di stati a due a due indistinguibili. I membri di due blocchi diversi sono sempre distinguibili. Se sostituiamo ogni blocco con un singolo stato, otteniamo un DFA equivalente, con un numero di stati non superiore a quello di ogni altro DFA per lo stesso linguaggio.
- **Quando una stringa distingue due stati?**
  Quando uno e uno solo tra gli stati è finale
- **Quando due stati p e q sono k-indistinguibili?**
  Se nessuna stringa di lunghezza inferiore o uguale a k distingue p da q.
- **Quando due stati sono indistinguibili?**

Se e solo se sono k-indistinguibili per tutti i $k >= 0$.

- **Cos'è un automa finito deterministico? Quando riconosce una stringa x?**

E' un **automa a stati finiti** dove per ogni coppia di stato e simbolo in ingresso c'è una ed una sola transizione allo stato successivo.
Riconosce x quando l'ultima delle transizioni relative alla stringa, conduce ad uno stato finale.
F]

- **Cos'è un automa finito non deterministico? Quando riconosce una stringa x?**

E' una macchina a stati finiti dove per ogni coppia stato-simbolo in input possono esservi più stati di

destinazione.
Riconosce x quando l'ultima delle transizioni relative alla stringa, conduce ad uno stato finale.
[se $\delta^*(q0,x) \cap F \neq \mathbf{\Phi}$]

- **Cos'è un automa finito con mosse spontanee?**

Un automa che prevede delle funzioni di transizione non legate alla presenza o meno di particolari simboli in input, ovvero prive di vincoli riferiti ai caratteri letti.

Slide 8

- **Saper passare tra grammatiche unilineari destre e sinistre.**
  1) Costruire Automa della grammatica
  2) Ricavare automa riflesso
  3) Ricavare grammatica DX dall'automa riflesso
  4) "Rovesciare" i membri dx delle produzioni (es A-> bC diventa A-> Cb)
- **Automi finiti deterministici e linguaggi regolari.**

Il teorema di Kleene dimostra che la collezione dei linguaggi accettati da automi a stati finiti deterministici coincide sia con la collezione dei linguaggi regolari che con i linguaggi definiti da espressioni regolari.

Slide 9

- **Cos'è un automa a pila?** E' un NFA dotato di uno stack che puo essere utilizzato per memorizzare una stringa di lunghezza arbitraria; E' una settupla: $M = < Q, \Sigma , \Gamma , \delta , q0, Z0, F >$ tale che: Q è l'insieme degli stati dell'unità di controllo , Σ è l'alfabeto di ingresso, Γ è Q è l'insieme degli stati finali.
- **Accettazione negli automi a pila:** un PDA segnala l'accettazione in due modi al ternativi: entrando in uno stato accettante oppure vuotando lo stack. I due metodi sono equivalenti, nel senso che qualunque linguaggio venga accettato tramite un metodo è accettato anche tramite l'altro (da un altro PDA).
- **Cos'è un automa a pila deterministico? L**a differenza di quello NON deterministico, per ogni terna [Stato, Simbolo In Input, Simbolo Sullo stack] definisce una sola possibile transizione, e non offre mai la scelta tra una mossa spontanea e una mossa con lettura.
- **Accettazione negli automi a pila deterministici:** i due modi di accettazione, per stato finale e per stack vuoto, nei DPDA non hanno la stessa portata. I linguaggi accettati per stack vuoto coincidono con quelli accettati per stato finale che hanno la proprietà di prefisso: nessuna stringa nel linguaggio è il prefisso di un'altra parola nel linguaggio stesso.
- **Automi finiti deterministici e grammatiche non ambigue.**

Se un linguaggio è riconosciuto da un automa push down [a pila] deterministico, esiste una grammatica non ambigua che lo genera. Non è vero il viceversa!

Slide 10

- **Proprietà di chiusura dei linguaggi liberi**
  La famiglia dei linguaggi liberi dal contesto è chiusa per la concatenazione e l'unione ma non per l'intersezione o la differenza.

Domande di LFT – Parte II

Slide 11

- **Che cos'è l'analizzatore sintattico, quali classi abbiamo e descriverle.**
  E' la componente del compilatore che si occupa di leggere la stringa sorgente e – se appartiene al linguaggio della grammatica – ne restituisce una derivazione o un albero sintattico, altrimenti si ferma segnalando l'errore.
  Le classi sono due:
  -discendenti [o top down]:
  Costruiscono un albero di parsificazione iniziando dalla radice e creando i nodi in preordine. Un analizzatore discendente espande le parti sinistre delle regole della grammatica nelle corrispondenti parti destre.
  -ascendenti [o bottom up]:
  Costruiscono l'albero dalle foglie alla radice.

Un analizzatore discendente riduce le parti destre delle regole della grammatica nei corrispondenti non terminali.

- **Chi riconosce i linguaggi (context free/...)?**

I riconoscitori dei linguaggi context free sono gli automi non deterministici

- **Quando un linguaggio è LL(k) / LR(k)?**

LL (k) → Quando è definito da una grammatica LL(k) per un valore finito di k >=

- **Indicare un linguaggio LL(1) ma non LR(0).**
  [$a^n b^m$ / 0<n<m]
- **Indicare un linguaggio non LL(1) ma LR(0).**

[$a^n b^m$ / n>m>0]

Slide 12

- **Definire l'insieme degli inizi di α.**

Insieme dei terminali con cui iniziano le stringhe derivabili da **α**

- **Definire l'insieme dei seguiti.**

Insieme dei terminali con cui iniziano le stringhe che seguono A nelle derivazioni di una grammatica.

- **Definite l'insieme guida.**

Insieme dei terminali con cui iniziano le stringhe generate a partire dalla produzione stessa

- **Quando una grammatica è LL(1)?**

Quando per ogni non terminale A e per ogni coppia di produzioni A-> **α** e A → **β,** insiemi guida sono disgiunti

- **Definire l'automa top down con prospezione 1.**

Un automa con un solo stato, che accetta per stack vuoto e con la capacità di leggere un simbolo in input senza spostare la testina.

Slide 13

- **Cos'è una ricorsione sinistra?**

E' una produzione che vede la presenza di un non terminale al lato sinistro della parte "destra" della produzione stessa.
Es. $A \rightarrow Aa$

- **Come si eliminano le ricorsioni sinistre immediate?**

Introducendo un nuovo non terminale ed inserendolo alla destra di tutte le produzioni del non terminale con ricorsione sinistra, eliminandolo poi dalla grammatica.

- **Come si eliminano le ricorsioni sinistre non immediate?**
- **Cos'è la fattorizzazione sinistra?**

La fattorizzazione sinistra serve ad eliminare un eventuale prefisso comune a due parti destre di regole associate allo stesso simbolo non terminale.

- **Come si effettua la fattorizzazione sinistra?**

y v | y w
yA'
v | w

Slide 14

- **Definizione di handle.**

E' una sottostringa che costituisce il corpo di una produzione, la cui riduzione rappresenta un passo della derivazione rightmost al contrario.

- **Quando si dice che una produzione è un handle?**

Quando è possibile riconoscere una regola che ha generato una specifica sequenza di simboli per ricostruirne una derivazione ascendente.
Es.: $S \rightarrow^* aAw \rightarrow aBw$, la produzione $A \rightarrow B$ [o solo B] nella posizione "dopo a" è un handle di aBw.

- **Cos'è un prefisso ascendente valido (completo)?**

Data una derivazione $S \rightarrow^* aAw \rightarrow aBw$, e dato $B=B_1B_2$, aB_1 è un prefisso ascendente valido.
E' completo perché B_2 vale **ε,** ovvero si è trovato un Handle.

- **Costruzione dell'automa riconoscitore dei prefissi ascendenti validi.**

1) Si aggiunge alla grammatica un nuovo assioma S' e la produzione $S' \rightarrow S$

2) Lo stato iniziale I_0 dell'automa è la chiusura di $\{S' \rightarrow \cdot S\}$
3) Si costruisce l'insieme degli stati QF e la funzione di transizione δF
[uno stato per ogni token che è possibile trovare in ogni produzione (facendo progressivamente avanzare il puntatore e analizzando ogni volta la nuova situazione)]

- **Cos'è un item?**

Un item per una grammatica G è una produzione di G con un punto in qualche posizione del suo membro destro.

- **Chiusura di un insieme di item: definizione e algoritmo.**

Se I è un insieme di Item per la grammatica G allora CLOSURE(I) è costruito seguendo 2 regole:
- In CLOSURE (I) vengono aggiunti tutti gli Item già presenti in I

- Se A -> α.Bβ è in CLOSURE(I) e B -> γ è una produzione allora si aggiunge a CLOSURE (I) anche l'item B ->.γ (se non c'è ancora). Tale regola si applica finchè si possono aggiungere elementi.

- **Quali tipi di stato abbiamo?**

- *Di spostamento [shift]* se contiene solo item della forma B ->x·γ

- *Di riduzione [reduce]* se contiene un unico item della forma $A \rightarrow B\cdot$
- *Inadeguato* se contiene item di entrambe le forme

- **Quando una grammatica è SLR(0)?**

Quando non ha stati inadeguati [con shift e riduzioni nello stesso stato]

- **Automa per il riconoscimento dei linguaggi SLR(0).**
- **Algoritmo per l'automa riconoscitore dei linguaggi SLR(0).**
- **Automa per il riconoscimento dei linguaggi SLR(1).**
- **Quando un linguaggio non è SLR(1)?**

Quando ha conflitti e stati inadeguati

Slide 15

- **Cos'è una SDD?**

Sono generalizzazioni delle grammatiche context-free in cui ad ogni simbolo della grammatica è associato un insieme di attributi.

- **Quali tipi di attributi abbiamo?**

Ereditati e sintetizzati

- **Cos'è l'albero di parsificazione annotato?**

E' un albero di parsificazione che mostra i valori degli attributi.

- **In quale ordine devono essere valutati gli attributi?**

Prima di valutare gli attributi di un nodo, bisogna valutare gli attributi dai quali dipende il suo valore.
In presenza di attributi sintetizzati (e SOLO sintetizzati), essi possono essere valutati in ordine bottom-up; in caso di presenza mista, non c'è garanzia che ci sia almeno un ordine in cui valutare gli attributi perché potrebbero essere presenti regole "circolari".

- **Come si costruisce il grafo delle dipendenze e come si valuta?**

- Per ogni nodo dell'albero di parsificazione, annoto a lato gli attributi del nodo stesso.

  - Per ogni nodo X a cui è applicata una produzione p, se alla stessa è associata una regola semantica che definisce un attributo sintetizzato in termini del valore dell'attributo b di un altro nodo Y, si crea un arco tra i due nodi interessati [da Y.b a X.c]; se a p è associata una regola semantica che definisce un attributo ereditato Z.c in termini del valore X.a, si crea un arco da X.a a Z.c.

Slide 16

- **Cos'è un SDT?**

E' una definizione guidata dalla sintassi in cui le azioni semantiche, racchiuse tra graffe, sono inserite nei membri destri delle produzioni, in posizione tale che il valore di un attributo sia disponibile quando un'azione fa ad esso riferimento.

- **Quali classi di SDD (e spiegarle) sono implementabili come SDT? Come?**

- SDD S- Attribuita [tutti gli attributi sono SINTETIZZATI]
  - SDD L-Attribuita [SINTETIZZATI ed EREDITATI]

  Trasformare una SDD L-Attribuita in una SDT:
  → Inserire le azioni che calcolano gli attributi ereditati per un non terminale A immediatamente prima dell'occorrenza di A nel corpo della produzione
  → Se diversi attributi ereditati per A dipendono uno dall'altro, ordinare la valutazione degli attributi in modo che quelli necessari prima siano calcolati per primi
  → Porre le azioni che calcolano un attributo sintetizzato per la variabile a SX in una produzione alla fine del corpo della produzione stessa

- **Associare le classi di SDD con le grammatiche LL/LR parsificabili.**
  Mi piace la foca che mangia e che gioca.

Slide 17

- **Valutazione top down di grammatiche L-attribuite.**

Ad ogni non terminale si associa una funzione che ha come parametri in input i valori degli attributi ereditati dalla variabile e restituisce i valori dei suoi attributi sintetizzati.
La funzione per un non terminale ha una variabile locale per ogni attributo ereditato o sinstetizzato per i simboli che compaiono nelle parti destre delle produzioni per quel non terminale.

- **Valutazione top down di grammatiche S-attribuite.**

Ma..hai mica sbagliato a scrivere? Mica volevi scrivere bottom-up?

Slide 18

- **Tradurre:**
  - **E → E + E | – E | (E) | id | num**
  - **P→S**

- **S → id = E ; | if (B) S | if (B) S else S | while (B) S | S S**
- **B → B || B | B && B | !B | (B) | E rel E | true | false**