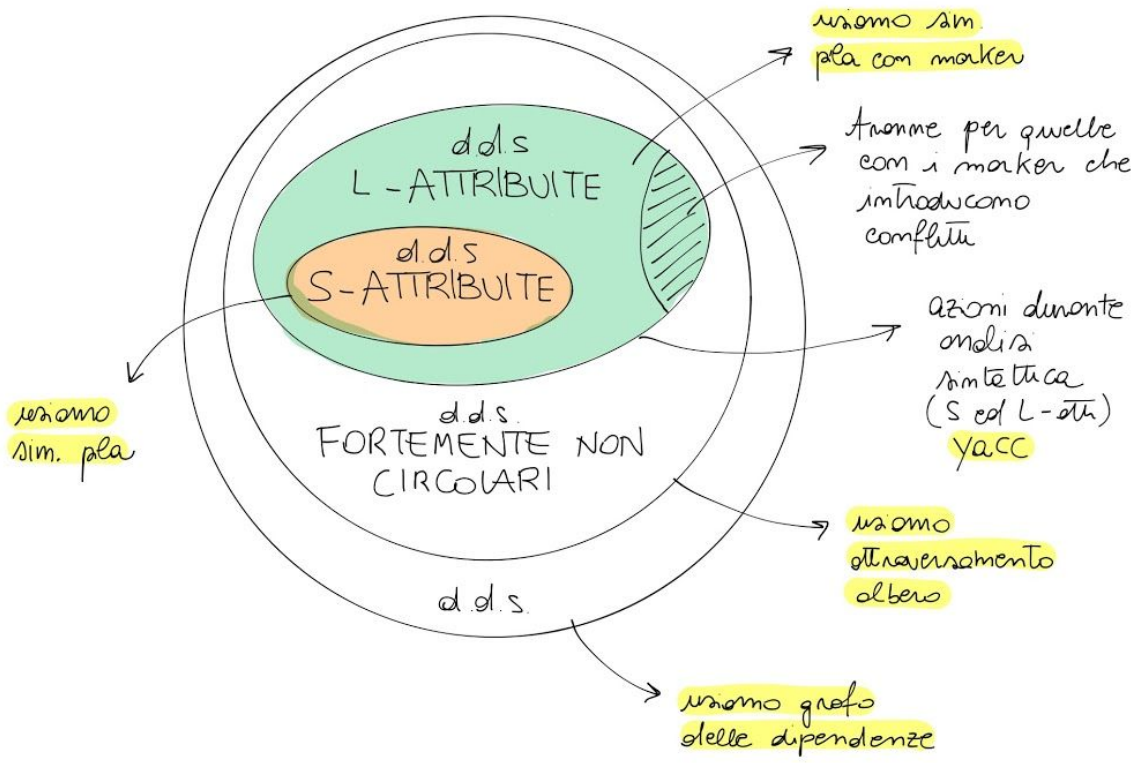
DOMANDE TEORIA FLT2

SCHEMA DDS



Esempi domande su traduzione diretta della sintassi

A) Descrivere il metodo del grafo delle dipendenze per il calcolo degli attributi. Considerando inoltre i seguenti metodi per il calcolo degli attributi:

1. ***Grafo delle dipendenze***
2. ***Valutazione di una definizione S-attribuita con azioni sulla pila degli attributi (azioni del tipo val[ntop] := ....)***
3. ***Valutazione di una definizione L-attribuita con introduzione di marker e azioni sulla pila degli attributi***

Dire, per ciascuna coppia di metodi, qual è più generale dell'altro (il metodo x è più generale del metodo y se in tutti i casi in cui si può applicare y si può applicare anche x, ma non viceversa) e perché. Indicare inoltre qual'è il vantaggio dei metodi meno generali (e quindi il motivo per cui non applicare sempre il metodo più generale).

Risposta:

Il metodo generale di valutazione di attributi serve per la valutazione ***dopo l'analisi sintattica*****.** Dato l'albero di derivazione, si costruisce il grafo delle dipendenze dove:

- **Nodi:** per ogni nodo n dell'albero di derivazione, per ogni attributo *a* del simbolo del nodo, si ha un nodo nel grafo delle dipendenze

- **Archi:** per ogni nodo interno n dell'albero di derivazione, per ogni azione b.=f(c1,...,ck) associata alla produzione utilizzata per il nodo n, si introducono gli archi orientati c1→ b,...ck→ b

La valutazione degli attributi può avvenire in qualunque ordine (totale) che rispetta questo ordinamento. Si tratta del cosiddetto ***ordinamento topologico*** di un grafo; un ordinamento totale n1<...<nk dei nodi tale che se vi è un arco ni → nj allora i<j. Ne esiste sempre almeno uno se il grafo non ha cicli.

Il metodo 1 è più generale del 3 che è più generale del 2. Il vantaggio dei metodi 2 e 3 rispetto all'1 consiste nel poter eseguire la valutazione degli attributi durante l'analisi sintattica evitando quindi il costo in spazio della memorizzazione di strutture dati come l'albero di derivazione e il grafo delle dipendenze e il costo in tempo dell'individuazione di un ordinamento topologico del grafo delle dipendenze. Il metodo 2 ha sul metodo 3 come unico vantaggio la semplicità concettuale.

B) Descrivere sinteticamente i metodi visti nel corso per effettuare la traduzione diretta dalla sintassi durante o dopo l'analisi sintattica, illustrando in particolare quali sono gli eventuali vantaggi/svantaggi/limitazioni nei vari casi.

Risposta:

I metodi per effettuare la traduzione visti nel corso sono:

1. **Traduzione durante l'analisi sintattica con memorizzazione degli attributi su pila;** le azioni sono effettuate in corrispondenza delle riduzioni di produzioni; gli attributi sintetizzati sono in posizione corrispondente a quella del simbolo nella pila dell'analisi sintattica; gli attributi ereditati sono in corrispondenza di simboli " marker ", introdotti per fare in modo che l'analizzatore sintattico esegua azioni durante l'analisi di una parte destra di produzione e dove si memorizzano gli ereditati per sapere dove sono quando servono. Si può applicare per le d.d.s L-attribuite e S-attribuite (tranne nei casi in cui l'introduzione dei marker introduce conflitti di analisi sintattica). La d.d.s può essere espressa anche come schema di traduzione, posizionando le azioni che calcolano attributi sintetizzati al fondo della parte destra della produzione, mentre le azioni che calcolano attributi ereditati vengono posizionate prima dell'analisi sintattica di quel determinato simbolo di cui è attributo ereditato.
2. **Traduzione dopo l'analisi sintattica con visita dell'albero di derivazione;** traduzione dopo l'analisi sintattica, con visita dell'albero di derivazione, limitante a dds fortemente non circolare. In una fase di analisi della grammatica si determina se il metodo può essere applicato. Il metodo può funzionare se esiste un ordine parziale degli attributi di ogni simbolo in modo che
    a. Il grafo delle dipendenze locale alla produzione, arricchiti con archi corrispondenti all'ordine per ciascun attributo sia aciclico.
    b. Se in tale grafo c'è un cammino da un attributo ad un altro, allora il primo precede il secondo nell'ordine

    L'aciclicità viene verificata solo localmente, Se valgono le condizioni precedenti allora la dds si dice fortemente non circolare. Quindi, si possono scrivere funzioni per calcolare ciascun attributo sintetizzato di un simbolo, da chiamare rispettando l'ordine e passando come parametri gli attributi ereditati precedenti nell'ordine. Se il grafo locale, arricchito con l'ordinamento corrente degli attributi, ha un ciclo, si esce con risultato negativo.
3. **Traduzione dopo l'analisi sintattica con il metodo del grafo delle dipendenze;** Per uno specifico input, dell'albero di derivazione di costruisce il grafo delle dipendenze, cioè un grafo nel quale gli

archi indicano delle dipendenze tra i vari attributi Se un attributo A.x dipende da un attributo A.y allora da A.y a A.x i sarà un arco diretto. Si eseguono le azioni in un ordine che rispetta il grafo delle dipendenze, cioè un ordine topologico del grafo.

L'ultimo metodo è il più generale, gli altri hanno le limitazioni indicate. Rispetto al 3, il metodo 2 ha il vantaggio di non dover memorizzare il grafo delle dipendenze totale e non dover calcolare un ordine di effettuazione delle operazioni per ogni specifico input. Rispetto al 2, il metodo 1 ha il vantaggio di non dover memorizzare l'albero di derivazione.

C) Indicare quale o quali dei metodi descritti nel corso è utilizzato, o sono utilizzati, da yacc, ed illustrare il significato del seguente frammento di codice:

A : B C D {$$=g($1,$3);}

indicando quando viene eseguita l'azione fra {}, in che cosa consiste tale azione e come sarebbe scritta in una definizione diretta dalla sintassi.

Risposta:

Yacc utilizza il metodo di valutazione con azioni su pila degli attributi, perciò accetta d.d.s. S attribuite ed L-attribuite.

A: B C D rappresenta la produzione A⟶ B C D mentre {$$=g($1,$3)} rappresenta l'azione per calcolare un attributo di A (A.s).Dato che è post al fondo, l'attributo A.s è sintetizzato, e si calcola tramite una funzione g che ha come parametri B.s, attributo sintetizzato di B, e D.s, attributo sintetizzato di D. L'azione viene eseguita al momento della riduzione della produzione. E in una d.d.s. sarebbe scritta in questo modo A⟶ B C D A.s:= g(B.s, D.s)

Spiegare il significato generale dell'attributo E.place in schemi di questo tipo, illustrando in particolare per le produzioni in questo schema e, nell'albero di derivazione costruito in precedenza, per le occorrenze di E che non generano un semplice identificatore

Spiegare il significato generale degli attributi true e false per il simbolo non terminale E in questo tipo di d.d.s.

Risposta:

Spiegare degli attributi E.truelist, E.falselist e S.nextlist

Risposta:

E.truelist è un attributo sintetizzato di E, che contiene la lista di istruzioni di salto incomplete, facenti parte della traduzione di E, la cui destinazione è da completare con l'indirizzo a cui saltare quando E è vera.

E.falselist è l'analogo per E falsa.

S.nextlist è un attributo sintetizzato di S, che contiene la lista di istruzioni di salto incomplete, facenti parte della traduzione di S, la cui destinazione è da completare con l'indirizzo a cui saltare quando l'esecuzione di S è terminata.

Facendo riferimento al metodo di traduzione delle espressioni booleane e delle istruzioni di controllo con il "metodo dei salti" e il "backpatching", supponiamo che la traduzione di una espressione

booleana E sia costituita dalle istruzioni di codice intermedio da quella di indirizzo m a quella di indirizzo n. Motivando la risposta, indicare quale delle 3 affermazioni seguenti è corretta per l'esecuzione del codice intermedio in una macchina virtuale:

1. **dopo aver eseguito l'istruzione di indirizzo n si arriva sempre ad eseguire l'istruzione di indirizzo n+1 per incremento del Program Counter (non per un goto condizionato o incondizionato)**
2. **dopo aver eseguito l'istruzione di indirizzo n non si arriva mai ad eseguire l'istruzione di indirizzo n+1 per incremento del Program Counter**
3. **a seconda della forma dell'espressione E e del suo valore (a run time) è possibile che dopo aver eseguito l'istruzione di indirizzo n si arrivi ad eseguire l'istruzione di indirizzo n+1 per incremento del Program Counter**

Risposta:

La risposta corretta è la 2. Con questo metodo, dal codice che costituisce la traduzione di E si esce sempre con un salto: una delle istruzioni di salto i cui indirizzi sono contenuti in E.truelist ed E.falselist, e le cui destinazioni vengono riempite con il backpatching con indirizzi che dipendono dal contesto in cui E occorre.
É possibile che una delle due destinazioni sia l'istruzione n+1, ed è possibile che uno dei salti la cui destinazione viene riempita con n+1 si trovi all'indirizzo n, cioè è possibile che si esegua l'istruzione n e subito dopo l'istruzione n+1, ad esempio per E "a<b" in "a<b or c<d" la traduzione è:
100 if a<b goto _
101 goto 102
102 if c<d goto _
103 goto _
per E "a<b" si ha m=100, n=101 e l'istruzione n salta a n+1 (102). Tuttavia, non si tratta di incremento del PC, ma di assegnazione al PC della destinazione del salto, che incidentalmente ottiene lo stesso risultato.

Indicare in che senso uno schema di traduzione specifica l'ordine in cui eseguire le azioni associate alle produzioni.

Risposta:

Uno schema di traduzione specifica un ordine delle azione come segue, se fatto dopo l'analisi sintattica:

- si aggiungono all'albero i nodi dell'azione
- l'ordine delle azioni è quello in cui i nodi corrispondenti sono vistati nel solito topo di attraversamento

Uno schema di traduzione non specifica solo l'ordine in cui eseguire le azioni ma anche il momento in cui eseguire.
Esempio: A⟶ B{a0} C {a1} D indica che l'azione a0 va eseguita prima di visitare il sottoalbero C ma dopo quello di B e che l'azione A1 va eseguita dopo aver visitato il sottoalbero di C ma prima di quello di D.

Illustrare la sequenza di passi e configurazioni della memoria (record di attivazione, valori dei parametri etc) per l'esecuzione dell'istruzione:

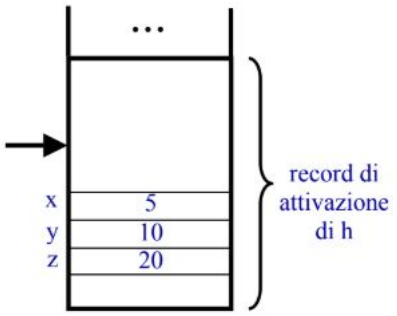
x := f(g(y),z)

Si supponga che:

- **l'istruzione faccia parte del codice di una funzione h**
- **x, y e z siano variabili locali di h e al momento dell'esecuzione dell'istruzione valgono rispettivamente 5, 10 e 20**

- **il valore di g(10) sia 30**
- **il valore di f(30,20) sia 50**

Pertanto la prima configurazione della memoria può essere indicata come segue:



I passaggi (senza i disegni corrispondenti) sono:

1. valutazione del parametro attuale y e passaggio del valore 10;
2. chiamata della funzione g con conseguente allocazione del suo record di attivazione;
3. esecuzione della funzione g con computazione del valore da restituire, 30;
4. uscita dalla funzione g con restituzione del valore 30 e deallocazione del record di attivazione di g;
5. passaggio del valore 30;

(nota: tutti i passaggi precedenti costituiscono la valutazione del parametro attuale g(y) per la chiamata di f)

6. valutazione del parametro attuale z;
7. passaggio del valore 20;
8. chiamata della funzione f con conseguente allocazione del suo record di attivazione;
9. esecuzione della funzione f con computazione del valore da restituire 50;
10. uscita dalla funzione f con restituzione del valore 50 e deallocazione del record di attivazione di f;
11. assegnazione del valore restituito, 50, alla variabile x.

Nota: tutto quanto sopra avviene supponendo, come precisato in occasione dell'esame, che il meccanismo di passaggio parametri sia quello per valore, l'unico di cui si è trattato nel corso. Alcuni hanno risposto che h chiama f, che a sua volta chiama g(y)... si noti che implementare un meccanismo del genere sarebbe molto più complicato. Si tratta essenzialmente del meccanismo di chiamata "per nome" menzionato sul testo, che ha prevalentemente interesse storico, esisteva nell'Algol 60 (nel senso di 1960...), e su Internet si trova ad esempio menzionato cosi: "Alan Perlis, who was on the Algol60 committee, told me that call-by-name was a mistake..." "Algol 60 had the irritating misfeature of CallByName arguments;... This was so annoying to compiler writers and users both that no other language to my knowledge has followed Algol's questionable lead here"

Illustrare in che senso una istruzione di codice intermedio del tipo utilizzato nel corso per la traduzione di accesso ad elementi di array:

a := b[c]

differisce dall analoga istruzione del tipo:

x := y[z]

del linguaggio ad alto livello, e come una istruzione come la seconda può essere tradotta utilizzando le istruzioni del codice intermedio introdotte nel corso.

Risposta:

Le istruzioni differiscono perché:

- nel linguaggio intermedio b[c] corrisponde all'indirizzo individuato da b, modificato sommandovi, in bytes, il valore che si trova all'indirizzo individuato da c; inoltre, gli operandi "b" e "c" possono essere indirizzi, o puntatori ad elementi della tabella dei simboli che descrivono gli oggetti

- nel linguaggio ad alto livello y[z] corrisponde all'elemento z-esimo dell'array y, e quindi il suo indirizzo si ottiene a partire da quello a partire dal quale è memorizzato l'array y, sommandovi (z-inf)*w dove "inf" è l'indice del primo elemento effettivo dell'array (nel linguaggio C si ha peraltro sempre inf=0), e "w" è la dimensione degli elementi dell'array

Pertanto, nel caso del C (inf=0) l'istruzione può essere tradotta con:
t1 := z*w
x := y[t1]

(in realtà con gli schemi visti nel corso al posto della seconda istruzione se ne ottiene una che assegna y[t1] a un altro temporaneo t2, e una che assegna t2 a x) questo va bene anche nel caso generale purché nell'elemento della tabella dei simboli che descrive l'array y venga messo come indirizzo non quello (addry) a partire dal quale è effettivamente memorizzato il primo elemento effettivo dell'array y, ma l'indirizzo: addry - inf*w

il significato generale dell'attributo E.place in schemi di questo tipo, illustrando in particolare per le produzioni in questo schema e, nell'albero di derivazione costruito in precedenza, per le occorrenze di E che non generano un semplice identificatore

Risposta:

Spiegare inoltre, per l'azione associata alla produzione per il while, il ruolo dell'attributo S1.next list, della istruzione goto emessa, ed indicare quali (uno, l'altra, entrambi, nessuno) di questi è effettivamente necessario nel caso particolare della traduzione dell'istruzione while a < b or a < c do a := a+d.

Risposta:

La destinazione della istruzione 103 viene completata nel momento in cui sia noto dove deve fluire il controllo dell'esecuzione quando è terminata l'esecuzione dell'intera istruzione while. Il goto (qui istruzione 106, necessaria) serve a fare in modo che una volta terminato il corpo del ciclo si torni alla valutazione della condizione.
S1.nextlist in questo caso non è necessario, ma in generale serve a fare in modo che se nel corpo del ciclo ci sono dei salti che devono portare al di fuori dello stesso, si salti direttamente alla valutazione della condizione e non al goto di cui sopra.

Dare la definizione di "definizione diretta dalla sintassi L-attribuita" e lo scopo per cui si introduce tale definizione.

Risposta:

E' L-attribuita una definizione diretta dalla sintassi se ogni attributo calcolato in un'azione associata ad una produzione
A ⟶ X1 ... Xn è un attributo sintetizzato, oppure è un attributo ereditato di Xj che dipende solo da:

- attributi di X1... Xj-1
- attributi ereditati di A

Lo scopo per cui si introduce è per risolvere i problemi di analisi sintattica per gli attributi ereditati. Questi infatti non devono dipendere da attributi di simboli successivi o attributi sintetizzati del

simbolo genitore poiché questi verranno generati in seguito. Ponendo queste limitazioni è possibile determinare le azioni durante l'analisi sintattica

Spiegare in generale per che cosa vengono usati gli attributi truelist e nextlist nel metodo di traduzione delle espressioni booleane che li utilizza.
Risposta:

Descrivere il metodo di traduzione diretta dalla sintassi che utilizza il grafo delle dipendenze. Se per una data definizione diretta dalla sintassi e una data stringa si può utilizzare tale metodo, è anche possibile effettuare la traduzione con yacc? Motivare la risposta.
Risposta:
vedere il materiale del corso per la descrizione del metodo. Nei casi in cui si può utilizzare, non è detto sia possibile effettuare la traduzione con yacc, che (oltre a porre requisiti dal punto di vista sintattico) richiede che la dds sia L-attribuita, dovendo effettuare le azioni durante l'analisi sintattica.

Spiegare che cos'è un attributo. Descrivere come e perché si distingue tra attributi sintetizzati ed ereditati.
Risposta:
un attributo di un simbolo terminale o non terminale è un'informazione associata ad ogni occorrenza del simbolo nell'albero di derivazione di una sequenza di simboli, informazione che serve per la traduzione.
Come si distingue:

- un attributo di un simbolo A è sintetizzato se tutte le azioni che lo calcolano sono associate a produzioni in cui l'occorrenza del simbolo A, per la quale l'attributo viene calcolato, occorre a sinistra nella produzione.
- un attributo di un simbolo A è ereditato se tutte le azioni che lo calcolano sono associate a produzioni in cui l'occorrenza del simbolo A, per la quale l'attributo viene calcolato, occorre a destra nella produzione.

Nel progettare una definizione diretta dalla sintassi è utile distinguere tra attributi sintetizzati ed ereditati perché con i primi si rappresenta informazione che dipende anche o solo dalla parte di input derivata dal simbolo, con i secondi si rappresenta informazione che dipende dal contesto, cioè dalla parte di input precedente e/o seguente quella derivata dal simbolo. La distinzione è utile per organizzare il calcolo degli attributi:

- durante l'analisi sintattica, introducendo la restrizione a dds L-attribuite (che fa riferimento alle due classi di attributi) e prevedendo momenti diversi, rispetto alle operazioni di analisi sintattica, per gli attributi ereditati e sintetizzati di uno stesso simbolo
- dopo l'analisi sintattica, con il metodo dell'attraversamento dell'albero di derivazione, che prevede di calcolare un attributo sintetizzato "s" di A visitando il sottoalbero con radice A, e dopo aver calcolato gli eventuali attributi ereditati di A da cui A.s può dipendere

Con yacc è possibile generare un traduttore che utilizza il metodo del grafo delle dipendenze? Illustrare poi il significato del seguente frammento di codice: A : B C D { $$ = g($2,$3);} ; indicando fra l'altro quando viene eseguita l'azione fra {} nel programma generato da yacc, e come verrebbe scritta tale azione in una definizione diretta dalla sintassi.

Risposta:

non è possibile, yacc genera un traduttore che calcola gli attributi durante l'analisi sintattica con il metodo visto nel corso, memorizzando gli attributi su pila, e richiede che la dds sia Lattribuita.

Il frammento rappresenta la produzione (unica produzione per il simbolo A): A BCD L'azione va considerata come in uno schema di traduzione, quindi essendo al fondo della parte destra della produzione viene eseguita quando la produzione viene ridotta, calcola l'attributo sintetizzato (unico) di A utilizzando quelli di C e D, in una dds verrebbe quindi scritta: A.s := g(C.s,D.s)

Spiegare il significato degli attributi true e false nella traduzione delle espressioni booleane con il metodo dei salti e indirizzi simbolici, indicando anche se si tratta di attributi sintetizzati o ereditati.

Risposta:

sono attributi ereditati di una espressione booleana E e rappresentano l'etichetta (indirizzo simbolico) a cui saltare quando l'espressione è vera, nel caso di E.true, è falsa nel caso di E.false. Tale etichetta è determinata dal contesto in cui E occorre.

Se è possibile usare il metodo del grafo delle dipendenze non è detto che si possano eseguire le azioni durante l'analisi sintattica: è necessario che la dds sia L-attribuita.

Se una dds può essere realizzato in yacc, è L-attribuita perchè YACC usa il metodo di esecuzione delle azioni durante l'analisi sintattica. Per tutte le dds L-attribuite si può anche usare il metodo dell'attraversamento dell'albero di derivazione, utilizzando l'attraversamento in profondità da sinistra a destra.

Dare la definizione di "definizione diretta dalla sintassi attribuita" e di "definizione diretta dalla sintassi L-attribuita" ed illustrare come e perché le azioni di tali definizioni possono essere realizzate durante l'analisi sintattica ed espresse in termini della pila degli attributi.

Risposta:

Una definizione diretta dalla sintassi che ha solo attributi sintetizzati si dice S-attribuita. In questo caso non è un problema trovare un ordine di valutazione degli attributi: va bene qualunque ordine "bottom-up" che cioè valuta gli attributi di un nodo dell'albero di derivazione dopo aver valutato gli attributi dei figli di quel nodo**.** Idea: effettuiamo l'azione che calcola A.s quando viene effettuata la riduzione di A → X1... X

L'ordine delle azioni così ottenuto corrisponde esattamente al solito attraversamento dell'albero di derivazione. Quindi, l'analisi sintattica corrisponde ad attraversare l'albero e nel mentre potremmo quindi realizzare le azioni (????).

In particolare va bene questo, che ha il vantaggio di corrispondere esattamente all'ordine delle riduzioni quindi possiamo fare a meno di costruire l'albero.

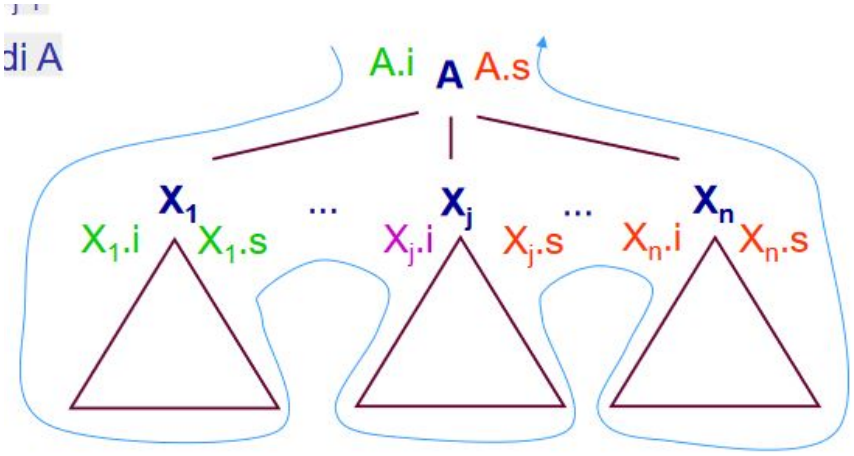
Per quanto riguarda una d.d.s L-attribuita.

Se vogliamo effettuare la valutazione durante l'analisi sintattica, dobbiamo restringere l'uso degli attributi ereditati, per rappresentare soltanto l'informazione che dipende dal contesto precedente al simbolo.

Per esempio: B → A Y

Per calcolare l'attributo ereditato di A prima di effettuare l'analisi sintattica di A, allora questo attributo ereditato A.i non deve dipendere da attributi di Y, né da attributi sintetizzati di B (vogliamo calcolarli dopo).
Una d.d.s. è L-attribuita, invece, se ogni attributo calcolato in un'azione associata ad una produzione $A \rightarrow X1 \ldots Xn$ è un attributo sintetizzato, oppure è un attributo ereditato di Xj che dipende solo da:

- attributi di X1... Xj-1
- attributi ereditati di A



Per specificare le azioni durante l'analisi sintattica (schema di traduzione) di una d.d.s. L-attribuita è necessario che: un'azione che calcola Xj.i andrà piazzata prima di Xj, e non dovrà fare riferimento ad attributi di simboli che occorrono alla sua destra; per non essere più restrittivi del necessario piazziamo quindi l'azione subito prima di X. Un'azione che calcola A.s, invece, potrà sempre essere piazzata al fondo, come nel caso di soli attributi sintetizzati