

# Esercizi modelli



## ► Esercizio Grammatica

DATA LA GRAMMATICA

$$S \rightarrow () \mid SS \mid (S)$$

VEDERE LE PRODUZIONI PER OBTENERE  $(((())))$ .

$S \Rightarrow SS \Rightarrow S(S) \Rightarrow S((S)) \Rightarrow S((SS)) \Rightarrow ((((()))$ ) ED ABBIAMO QUINDI OBTENUTO TALE STRUNGA DEL LINGUAGGIO.

## ► Esercizio Grammatica

DATA LA GRAMMATICA:

$$E \rightarrow E + T \mid E - T \mid T$$

$$T \rightarrow T * F \mid T / F \mid F$$

$$F \rightarrow a \mid (E)$$

QUINDI CON  $a$  UNICA VARIABILE DELL'ESPRESSIONE.

SCRIVERE UNA GRAMMATICA CON IN PIÙ: RADICE, POTENZA, CAMBIO SEGNO.

$$E \rightarrow E + T \mid E - T \mid T$$

$$T \rightarrow T * P \mid T / P \mid P$$

$$P \rightarrow -H \mid H$$

$$H \rightarrow FF \mid F \wedge F \mid F$$

$$F \rightarrow a \mid (E)$$

IN QUESTO MODO OTTERGO CHE LE PRECEDUTE VENGONO RISPETTATE. INFATTI VOGLIO AD ESEMPIO:

$$E \Rightarrow E + T \Rightarrow E + T * P \Rightarrow E + T * -H \Rightarrow E + T * -F \wedge F \Rightarrow E + T * -(E) \wedge (E) \dots$$

LA PRECEDUTA È RISPETTATA PERCHÉ  $T$  È DATO DAL PRODOTTO  $T * P$  QUINN'PER RIPIRARE LA FORMA PRIMA DOVUO SAPERE IL PRODOTTO

ANCORA PRIMA VALUTO  $F$   
ANCORA PRIMA VALUTO LE SINGOLE ESPRESSIONI

QUESTO VOL' DIA CHE PRIMA CALCOLO  $-H$   
E Poi faccio il prodotto infatti per ottenere  $P$  DOVO VALUTARNE  $-H$ .

## ► Esercizio Grammatica

SCRIVERE LA GRAMMATICA DI

$$L = \{ a^n b^n \mid n \geq 1 \}$$

$$S \rightarrow aSb \mid ab$$

INFATTI

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaaabb$$

## ► Esercizio Grammatica

SCRIVERE LA GRAMMATICA DI

$$L = \{ a^n b \mid n \geq 0 \text{ E PARI} \}$$

Sono strunque del linguaggio:

$b, aab, aaaaab, \dots$

$$S \rightarrow b \mid Ab$$

$$A \rightarrow Aab \mid ab$$

INFATTI

$$S \Rightarrow Ab \Rightarrow Aab \Rightarrow aaaaab$$

## ► ESEMPIO GRAMMATICA

SCRIVERE LA GRAMMATICA DI

$$L = \{ a^n b^n c^n \mid n \geq 1 \}$$

SONO STRANGE DEL LINGUAGGIO:

abc, aabbcc, ...

$$S \rightarrow a b c \mid a b c A B C$$

$$d A \rightarrow d d$$

$$b B \rightarrow b b$$

$$c C \rightarrow c c$$

$$c A \rightarrow A c$$

$$b A \rightarrow A b$$

$$c B \rightarrow B c$$

$$A B C \rightarrow A B C \mid \epsilon$$

INFATI:

$$\begin{aligned} S &\Rightarrow a b c A B C \Rightarrow a b c A B C A B C \Rightarrow a b c A B C A B C A B C \Rightarrow a b c A B C A B C \Rightarrow d b A c B c A B C \Rightarrow d A b c B c A B C \Rightarrow d A b B c C A B C \Rightarrow \\ &\Rightarrow d A b B c C A B C \Rightarrow d d b b c c A B C \Rightarrow d d b b c A c B C \dots \Rightarrow d d d b b b c c c \end{aligned}$$

## ► ESEMPIO GRAMMATICA

SCRIVERE LA GRAMMATICA DI

$$L = \{ a^n b^m c^{n+m} \mid n \geq 1, m \geq 0 \}$$

SONO STRANGE DEL LINGUAGGIO:

dc, aabbccc, aabbcccc, aabbccccc ...

$$S \rightarrow a B c \mid a S c$$

$$B \rightarrow b B c \mid b c \mid \epsilon$$

INFATI

$$S \Rightarrow a S c \Rightarrow a a S c c \Rightarrow a a a S c c c \Rightarrow a a a a B c c c c \Rightarrow a a a a b B c c c c c \Rightarrow a a a a b c c c c c$$

## ► ESEMPIO GRAMMATICA

SCRIVERE LA GRAMMATICA DI:

$$L = \{ a^{2^n} \mid n \geq 0 \}$$

SONO STRANGE DEL LINGUAGGIO:

a, aa, aaaa, aaaaaaaaa, ...

QUINDI HO LA COPPIA aa RIPETUTA n VOLTE.

$S \Rightarrow d | dd | N dd B$

$N \rightarrow NN | A$

Add  $\rightarrow$  add A

Aaa B → aaaa B | aaaa

TALE GRAMMATICA È STATA OTTENUTA CONSIDERANDO CHE :

$d^8 = dddd \quad dddd \quad$  cioè è composto da 2 pezzi da 4

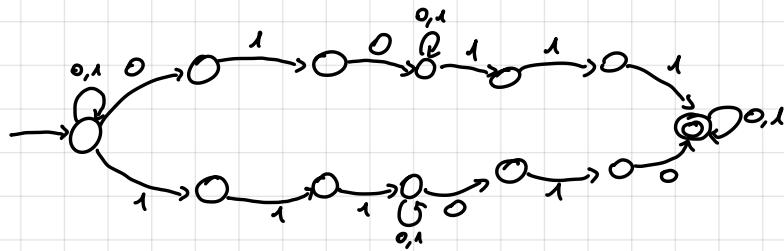
PER OTTENERE  $\delta^{16}$  AVRÓ 3 PARTI DA 8 OGNI UNA FORMATA DA 2 PEZZI DA 4 OTTENUTI AGGIUNGENDO 2 D PER OGNI COPPIA DI 3 PRIMA PRESENTE.

INFATTI:

$S \Rightarrow N \text{aa } B \Rightarrow NN \text{aa } B \Rightarrow NNN \text{aa } B \Rightarrow NNA \text{aa } B \Rightarrow NN \text{aaa } B \Rightarrow N \text{Aaaaa } B \Rightarrow Naaaa \text{Aaa } B \Rightarrow Naaaaaaa \text{B} \Rightarrow aaaaa \text{A aa aa dd } B \Rightarrow aaaaa \text{A aa dd } B \Rightarrow aaaaa \text{A dd dd } B \Rightarrow aad dd dd dd dd dd dd$

► ESAME MODELLI 13 GIUGNO 2019 A

G. AUTOMA CHE RICONOSCE STRANIE CONTENUTI LE SEQUELENZA O10 OPPURE 111



## H. GRAMMATICA CONTEXT FREE CUE GENERA

$$L = \left\{ a^n b^m \mid m \geq 0, n \geq 0, m > n \right\}$$

LA GRAMMATICA  $G = \langle \{a, b\}, \{\Delta, B, S\}, S, P \rangle$  È QUELLA CERCATA.

PÉIL SEGUENTIE WAGEN:

$$A \rightarrow a / a A B$$

$$\beta \rightarrow b | \beta \beta$$

## I. DATA /A GRAMMATICA

$$S \rightarrow aAb \mid aSb$$

$$A \rightarrow a A b b | a b$$

1) PER OBTENERE  $\frac{d^3x}{dt^3} = b_3$  FACCIO LE SEGUENTI DERIVAZIONI:

$\Rightarrow$   $\exists \bar{z} \in \mathbb{C}$   $\exists \bar{w} \in \mathbb{C}$   $\forall \bar{x} \in \mathbb{C}$   $\forall \bar{y} \in \mathbb{C}$   $A(\bar{x}, \bar{y}) \Rightarrow A(\bar{x} + \bar{z}, \bar{y} + \bar{w})$

2) DESCRIVO IL LINGUAGGIO GENERATO:

$$L = \{ a^nb^n \mid n \geq 2 \}$$

$$3) \quad S \rightarrow aAb \mid a \in \{b\}$$

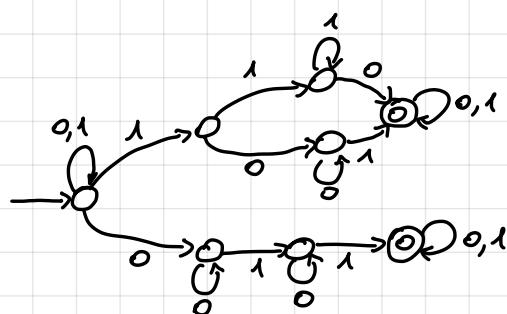
Value Scales Evaluate the Probability Now via GARCH Distributions in Quantitative

LA PRODUZIONE  $\alpha A \beta$  PUÒ ESSERE ELIMINATA PERCHÉ PUÒESSERE GIÀ OTTENUTA CON LE SEGUENTI DERIVAZIONI:

$S \Rightarrow \alpha S \beta \Rightarrow \alpha S \beta \beta$  SENZA L'AUTOLIO DI ULTERIORI PRODUZIONI

► ESAME MODELLI 13 GIUGNO 2019 B

G.



H. DESCRIVERE UNA GRAMMATICA CONTEXT-FREE CHE GENERA

$$L = \{ 0^n 1^m \mid n \geq 0, m \geq 0, n \neq m \}$$

TALE GRAMMATICA È  $G = \langle \{0, 1\}, \{S, A\}, S, P \rangle$

Dove l'insieme  $P$  delle produzioni è il seguente:

$S \rightarrow OA$  RICORDA CHE POSSO AVERE SOLO TERMI

$$A \rightarrow 0A1 \mid 0A \mid 01 \mid 0$$

I. 1) LE DERIVAZIONI SONO:

$$S \Rightarrow 1A00 \Rightarrow 1111A0000 \Rightarrow 1111100000$$

2) IL LINGUAGGIO GENERATO È:

$$L = \{ i0^n \mid n \geq 3 \text{ E DISPARA} \}$$

INFATTI AGGIUNGO SEMPRE DUE 1 E DUE 0 MA CONCILIO CON 10

3)

$$S \rightarrow 11A00$$

$$A \rightarrow 11A00 \mid 10$$

È STATA ELIMINATA LA PRODUZIONE  $11A00$  PERCHÉ È ESATTAMENTE UGUALE ALLA PRODUZIONE  $11A00$ . INVECE DI CONTINUARE A PRODURRE SULLA  $S$  SI PRODUCE SULLA  $A$  OTTENENDO LO STESSO LINGUAGGIO.

► ESEMPIO GRAMMATICA

$$A \rightarrow bA \mid \alpha B \mid \beta$$

$$B \rightarrow \alpha A \mid \beta B \mid \gamma$$

OTTERE L'E.R. CORRISPONDENTE.

$$\left\{ \begin{array}{l} A: bA + \alpha B + \beta \\ B: \alpha A + \beta B + \gamma \end{array} \right. \rightarrow \text{CO} \bar{\epsilon} A \text{ PUÒ ESSERE UNA DELLE 3, LO STESSO PER } B.$$

DEVO CALCOLARE RISPETTO A B così da sostituirlo in A, per fare ciò devo risolvere l'iterazione presente.

$$B = aA + bB + d = bB + aA + d = b(aA + bB + d) + dA + d = b^*aA + bbB + ba + dA + d = b^*(aA + d)$$

Allora sostituisco in A

$$A = bA + ab^*(aA + d) + b \quad \text{DEVO ALLORA RAGGRUPPARE RISPETTO A COSÌ DA POTER RISOLVERE L'ITERAZIONE.}$$

$$A = bA + ab^*aA + ab^*d + b = (b + ab^*a)A + ab^*d + b \quad \text{Allora risolvo l'iterazione.}$$

$$A = (b + ab^*d)A + ab^*d + b = \boxed{(b + ab^*d)}^+$$

PERCHÉ HO  $ab^*d + b$  OPPURE LA STESSA MA ITERATA. INFATTI INIZIALMENTE POSSO SCEGLIERE SE FERMARMI CON  $b^*a + b$  OPPURE CONCATENARGLI UN'ALTRA UGUALE CON ANCORA LA A OPPURE CONCATENARGLI UN'ALTRA UGUALE SENZA LA A CONCUDENDO LA CONCATENAZIONE.

### ► Esercizi Grammatica

1) La struttura palindroma di lunghezza pari, voglio di mostrare che il linguaggio è strettamente di tipo 2.

Assumo di avere un automa che riconosce  $L_1$  di  $n$  stati. Prendo  $z \in L_1$  con  $|z| > n$ , ciò vuol dire che avrò dei c.c.

In particolare prendo  $|z| > 2n$ . Una struttura palindroma pari è simmetrica necessariamente rispetto al centro. Questo vuol dire che se volgessi scrivere  $z = uvw$ , applicando quindi la scomposizione descritta dal Pumping Lemma, troverei che per ottenere una struttura apparentemente ad  $L_1$  "pompando"  $v$  dovrai averlo così che  $v$  sia pari al centro di  $z$  ma con  $|v| > n$  il centro si trova nella struttura  $w$  dovendo mantenere  $|uv| \leq n$ .

Non potendo "pompato" e per ottenere altre strutture del linguaggio allora  $L_1$  non è regolare.

Va fornita allora una grammatica di tipo 2:

$$G = \langle \{a, b\}, \{S\}, S, P \rangle$$

Il cui insieme  $P$  delle produzioni è il seguente:

$$S \rightarrow aSa \mid bSb \mid aa \mid bb \rightarrow \text{È DI TIPO 2 INFATTI NON HO TERMINAL A SINISTRA DELLE PRODUZIONI ED A DESTRA HO PIÙ DI UN TERMINALE}$$

La grammatica mi genera strutture come

$$S \Rightarrow aSa \Rightarrow abAba \Rightarrow abababab$$

2) La struttura palindroma di lunghezza dispari, voglio di mostrare che il linguaggio è strettamente di tipo 2.

La dimostrazione del caso dispari è la stessa del caso pari perché una struttura palindroma è simmetrica rispetto al centro anche nel caso dispari. Presento ora una grammatica di tipo 2:

$$G = \langle \{a, b\}, \{S\}, S, P \rangle$$

Il cui insieme di produzioni è il seguente:

$$S \rightarrow aSa \mid bSb \mid a \mid b$$

3) L<sub>3</sub> STRANIE PALINDROME DI LUNGHEZZA QUALSIASI, VOGLIO DI MOSTRARE CHE IL LINGUAGGIO È STETTAMENTE DI TIPO 2.

LA DEMOSTRAZIONE È DATA DALL'UNIONE DEL CASO PARI E CASO DISPARI.

DEVO FORNIRE UNA GRAMMATICA DI TIPO 2:

$$G = \langle \{a, b\}, \{S\}, S, P \rangle$$

IL CUI INSIEME DI PRODUZIONI È IL SEGUENTE:

$$S \rightarrow aSa \mid bSb \mid aa \mid bb \mid a \mid b$$

### ► Esercizi Grammatica

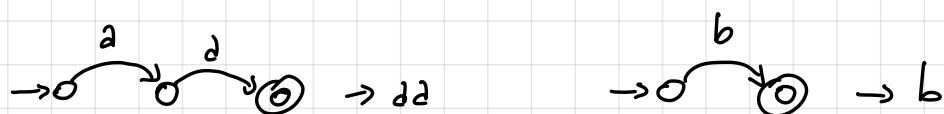
PASSAGGIO DA ER AD AUTOMA A GRAMMATICA AD ER:

1)  $L = \{ a^n b \mid n \geq 0 \text{ E PARI} \}$

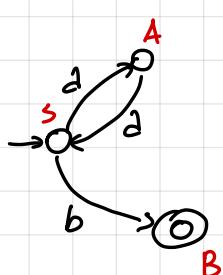
ER:

$$(aa)^* b$$

AUTOMA:



Allora



GRAMMATICA:

$$S \rightarrow aA \mid b$$

$$A \rightarrow aS$$

ESPRESSIONE REGOLARE

$$\begin{cases} S = aA + b \\ A = aS \end{cases} \Rightarrow S = aA + b = a(aS) + b = a^2S + b \Rightarrow (aa)^+b + b = (aa)^*b \text{ PERCHÉ L'ITERAZIONE SI CONCLIDE SEMPRE CON UNA } b.$$

2

$$S \rightarrow aS \mid bA \mid \epsilon$$

$$A \rightarrow aA \mid bS \mid \epsilon$$

DEVO LEVARE  $\epsilon$  DA A COSÌ DA RISPETTARE LA DEFINIZIONE PREMESSA. NELLE GRAMMATICHE DI TIPO 3 AVVIÀ  $\epsilon$  È OVUNQUE NON AUMENTA IL POTERE ESPRESSIVO DELLA GRAMMATICA.

Allora ho

$$S \rightarrow aS \mid bA \mid b \mid \epsilon$$

$$A \rightarrow aA \mid bS \mid a$$

ER:

$$\begin{cases} S = aS + bA + b + \epsilon \\ A = aA + bS + a \end{cases}$$

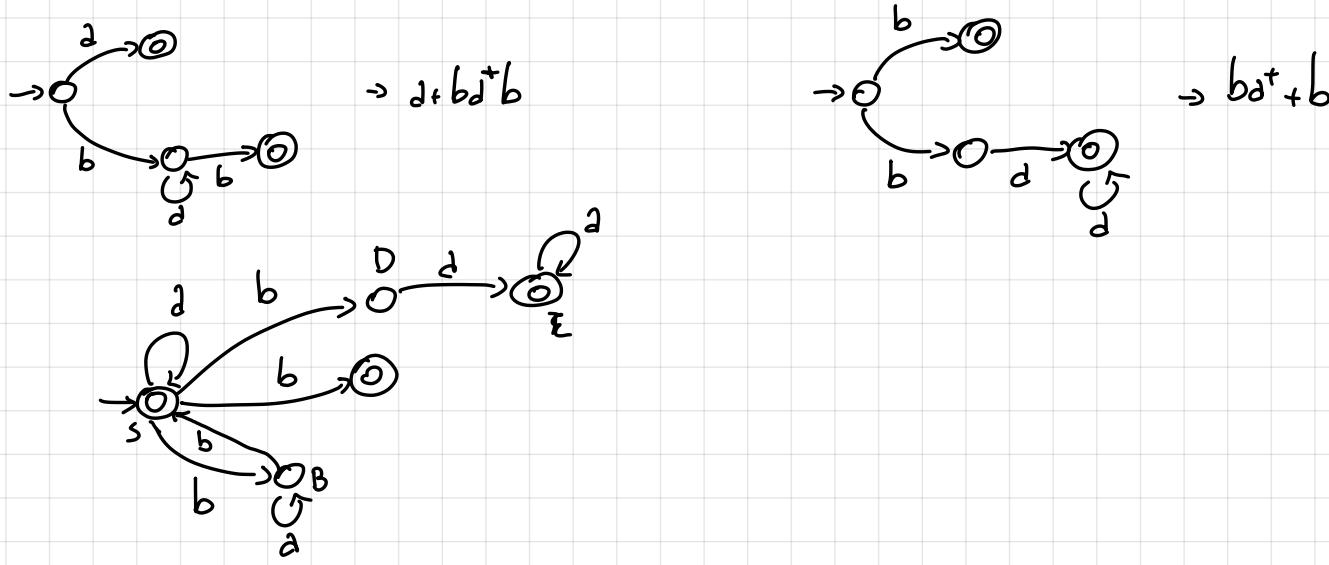
LAVORO SU A

$$A = aA + bS + a = a(aA + bS + a) + bS + a = aaA + abS + aa + bS + a = aaA + (ab + a)S + aa + a = a^t + a^*bS = a^*(bS + a)$$

SOSTITUISCO IN S

$$\begin{aligned} S &= aS + b(a^*(bS + a)) + b + \epsilon = aS + ba^*(bS + a) + b + \epsilon = aS + ba^*bS + ba^*a + b + \epsilon = (a + ba^*b)S + ba^*a + b + \epsilon = \\ &= (a + ba^*b)[(a + ba^*b)S + ba^*a + b + \epsilon] + ba^*a + b + \epsilon = (a + ba^*b)(a + ba^*b)S + (a + ba^*b)ba^*a + (a + ba^*b)b + (a + ba^*b) + \\ &ba^*a + b + \epsilon = (a + ba^*b)^*(ba^*a + b + \epsilon) = (a + ba^*b)^*(ba^*a + (a + ba^*b))^* \end{aligned}$$

AUTOMA:



GRAMMATICA:

$$S' \Rightarrow S \mid \epsilon \quad \text{RICORDA CAMBIO DI ASSORTE ALTRIMENTI NO E PRODUZIONI!}$$

$$S \Rightarrow aS \mid bB \mid bD \mid b \mid a$$

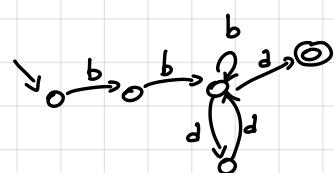
$$B \Rightarrow aB \mid bS$$

$$D \Rightarrow aE \mid a$$

$$E \Rightarrow a \mid aE$$

1)

1.1



1.2

$$S \rightarrow bB$$

$$B \rightarrow bK$$

$$K \rightarrow aA|bU$$

$$A \rightarrow aK$$

2)

$$S \rightarrow T\$$$

$$T \rightarrow (T)T|\epsilon$$

2.1

$$\text{SELECT}(S \rightarrow T\$) = \{ (\, \$ \}$$

$$\text{SELECT}(T \rightarrow (T)T) = \{ (\, ) \}$$

$$\text{SELECT}(T \rightarrow \epsilon) = \{ \, , \$ \}$$

SICCOME TUTTE LE INTERSEZIONI TRA I FIRST DI OGNI NON TERMINALE È VUOTA ALLORA G È LL(1).

2.2

HEGUO FADE SKILLE AD UNO PSEUDOCODICE SCRIVENDO LE CONDIZIONI NEGLI IF E CHIAMATE AI NON TERMINALI

PER SVOLGERE IL PARSING PREDITTIVO SCRIVO UNA FUNZIONE PER OGNI NON TERMINALE. IN QUESTO CASO AVRÒ QUINDI UNA FUNZIONE PER  $S$  ED UNA FUNZIONE PER  $T$ . IN OGNI FUNZIONE HO DEGLI IF I QUALI, ATTRAVERSO I SELECT E LA LETTURA DI 1 CARATTERE DELL'INPUT, POSSO DECIDERE QUALE PRODUZIONE EFFETTUARE. PER AVVIARE IL PARSER SI FA UNA PRIMA CHIAMATA SU  $S$  PER POI LEGGERE IL 1° CARATTERE DELL'INPUT.  
UNA VOLTA RICONOSCIUTA LA PRODUZIONE SI ENTRA NELL'IF E SI CHIAMA UN'ALTRA FUNZIONE SU UN NON TERMINALE OPPURE, SE È UNA PRODUZIONE FINACK (cioè NON CHIAMA SU NON TERMINALI) TERMINA L'ATTUALE CHIAMATA DI FUNZIONE. TORNANO A QUELLA DEL NON TERMINALE PRECEDENTE.  
IL RICONOSCIMENTO DA SUCCESSO QUANDO TUTTE LE CHIAMATE TERMINANO CON SUCCESSO. SE TUTTI GLI IF FALLISCONO FALLISCE ANCHE LA CHIAMATA E QUINDI TUTTO IL RICONOSCIMENTO.

4)

L'ALGORITMO DPLL VIENE USATO PER RISOLVERE IL PROBLEMA CNF E QUINDI TROVARE UN'ASSEGNAZIONE CHE SODDISFA UNA FORMULA CNF.

L'ALGORITMO DPLL È CARATTERIZZATO DA UNA RICORSIONE BINARIA DOVE SI ASSEGNA IL VALORE TRUE O FALSE AL LETTERALE  $x$ . COSÌ SI FA AGGIUNGENDO ALLA FORMULA  $F$  LA CLAUSOLA  $(x)$  o  $(\neg x)$  RISPECTIVAMENTE.

PRIMA DELLA RICORSIONE BINARIA PERÒ VENGONO APPLICATE EURISTICHE DI SEMPLIFICAZIONE DELLA FORMULA CHIAMATE:

- UNIT-PROPAGATION: ASSEGNO TRUE O FALSE AL LETTERALE E CANCELLA LE CLAUSOLE SODDISFATTE ED ELIMINA IL LETTERALE DALLE CLAUSOLE IN CUI VALE FALSE
- PURE-LITERAL ELIMINATION: SE CI SONO LETTERALI PURI CANCELLA LE CLAUSOLE IN CUI SONO PRESENTI

SE LA 1° CHIAMATA FALLISCE ALLORA SI EFFETUA LA CHIAMATA CON  $x = \text{FALSE}$ . SE FALLISCONO TUTTE E 2 LE CHIAMATE LA FORMULA È INSODDISFAZIONE.

IL COSTO COMPUTAZIONALE È DATO DALLE 2 CHIAMATE RICORSIVE E DALLE EURISTICHE:

$$T(n) = 2T(n-1) + h \quad \text{DOVE } h \text{ È IL NUMERO DI LETTERALI}$$

$$T(n-1) = 2T(n-2) + h$$

ALLORA

$$T(n) = 2^2 T(n-2) + 2h \quad \text{VEDIAMO QUINDI CHE IL COSTO È } \Theta(2^n) \quad \text{QUINDI ESPO-NENTRALE.}$$

All'inizio non trovo clausole unitarie e non ci sono letterali puri.

Assegno  $b = \text{TRUE}$  allora ottengo

$$(\neg b \vee d) \wedge (c \vee \neg d \vee e) \wedge (\neg b \vee \neg d \vee e) \wedge (c \vee d \vee e) \wedge (c \vee \neg d \vee e)$$

ho un letterale puro quindi si assegna  $c = \text{TRUE}$

$$(\neg b \vee d) \wedge (\neg b \vee \neg d \vee e)$$

ho un letterale puro e quindi pongo  $e = \text{TRUE}$

$$(\neg b \vee d)$$

Pongo  $b = \text{FALSE}$  e la computazione è conclusa.

5)

5.5 Non possiamo dedurre nulla

5.6 Possiamo dedurre che  $P \neq NP$

5.1 La classe  $P$  è l'insieme di tutti i linguaggi  $L$  per cui una MT det. decide in tempo polinomiale se  $x \in L$

5.2 La classe  $NP$  è l'insieme di tutti i linguaggi  $L$  per cui una MT non det. decide in tempo polinomiale se  $x \in L$ .

5.3

La classe  $NP$  è l'ins. dei linguaggi  $L$  per cui se  $x \in L$  allora esiste un certificato  $C(x)$  con le seguenti caratteristiche:

- $|C(x)|$  è polinomiale in  $|x|$
- una MT det. con in input  $x \in C(x)$  è in grado di verificare che  $x \in L$  in tempo polinomiale

5.4

Se vale la Def. 1 ed  $x \in L$  allora esiste almeno una computazione accettante di lunghezza polinomiale, allora tale computazione può essere rappresentata attraverso un certificato  $C(x)$  che sarà quindi di lunghezza polinomiale.

Se vale la Def. 2 allora si ha che  $x \in L$  e quindi avrò un ramo accettante il quale è descritto da  $C(x)$ . Con l'uso del non det. posso generare tutti i certificati e trovare  $C(x)$  in tempo polinomiale.

► ESAME 13-06-19

A)

$$T(n) \approx \frac{n(n+1)}{2} = \Theta(n^2)$$

Perché il 1° ciclo è svoltato  $n-1$  volte. Alla prima iterazione il 2° ciclo mi costa 1, alla 2° mi costa 2 e così via quindi avrò:

$$1 + 2 + 3 + \dots + n-1 \text{ e quindi è } \approx \frac{n(n+1)}{2} \text{ (somma dei primi } n \text{ numeri)}$$

Allora

$$T(10) \approx \frac{10(10+1)}{2} = 55$$

B) La classe  $P$  è l'insieme di tutti i linguaggi  $L$  che possono essere decisi da una MT det. in tempo polinomiale

La classe  $NP$  è l'insieme di tutti i linguaggi  $L$  che possono essere decisi da una MT non det. in tempo polinomiale

Si ha che  $P \subseteq NP$ , non si sa dire se  $P = NP$  o  $P \neq NP$

c)

PER DIMOSTRARE CHE P NON È PIÙ DIFFICILE DI Q POSSO AGIRE IN 2 MODI:

1. POSSO DEMONSTRARE CHE  $P \in NP$  SE  $Q \in NP$  OPPURE CHE  $P \in P$  SE  $Q \in P$ .
2. SE  $Q$  È NP COMPLETO OPPURE ESISTE UNA RIDUZIONE POLINOMIALE DI  $P$  A  $Q$  ALLORA BASTA PRESENTARE TALE RIDUZIONE

D)

**IMPORTANTE** → PROBLEMA SAT SI RIFERISCE SOLO ALLE FORMULE CNF

PER DIMOSTRARE CHE  $SAT \in NP$  MI BASTA MOSTRARE UN ALGORITMO CHE USA IL NON DET. CON COSTO POLINOMIALE.

ABBIANO PER OGNI LETTERALE 2 ASSEGNAZIONI CIÒ TRUE OPPURE FALSE QUINDI PER OGNI LETTERALE HO UNA COMPUTAZIONE IN WI È TRUE ED UNA IN WI È FALSE.

HO ALLORA UN GRADO DI NON DET. PARI A  $2^n$  ED OGNI COMPUTAZIONE È LUNGA  $n$ , È QUINDI POLINOMIALE.

ABBIANO QUINDI CHE  $SAT \in NP$ .

$\hookrightarrow n$  NUMERO DI LETTERALI

PER IL PROBLEMA 4-SAT LA DEMONSTRAZIONE È LA STESSA PERCHÉ ANCHE SE UNA CLAUSOLA HA 4 LETTERALI POSSO AVERE PIÙ CLAUSOLE.

E) UNA MT A 2 NASTRI HA LO STESSO POTERE COMPUTAZIONALE DI UNA AD 1 NASTRO, CIÒ CHE CAMBIA È IL TEMPO IMPIEGATO. ABBIANO INFATTI CHE UNA MT AD 1 NASTRO PUÒ SIMULARE UNA MT A 2 NASTRI IN TEMPO ESPONENTIALE.

UNA MT A 2 NASTRI POSSEDE 1 TESTINA PER NASTRO, CIÒ CHE CAMBIA È LA FUNZIONE DI TRANSIZIONE XA QUALSIASI STATO, OLTRE A DIRLE IL NUOVO STATO, DA UNA COPPIA DI CARATTERI DA SCRIVERE UNA SUL 1° ED UNA SUL 2° NASTRO E QUALE MOVIMENTO FAR PER OGNI TESTINA.

F) LA GERARQUIA CONSISTE IN 4 GRAMMATICHE:

· GRAMMATICA DI TIPO 3:

SONO TUTTE QUELLE GRAMMATICHE CARATTERIZZATE DA PRODUZIONI DEL TIPO

$$A \rightarrow \alpha B$$
$$A \rightarrow \alpha$$

CON  $A, B \in V_N$  ED  $\alpha \in V_T$ .

· GRAMMATICA DI TIPO 2:

SONO TUTTE QUELLE GRAMMATICHE CARATTERIZZATE DA PRODUZIONI DEL TIPO

$$A \rightarrow \beta$$

CON  $A \in V_T$  E  $\beta \in V^+$ .

· GRAMMATICA DI TIPO 1:

SONO TUTTE QUELLE GRAMMATICHE CARATTERIZZATE DA PRODUZIONI DEL TIPO

$$\alpha \rightarrow \beta$$

CON  $\alpha \in V^+ \cap V_T$  E  $\beta \in V^+$ .  $\rightarrow$  IN PIÙ HA  $|\alpha| \leq |\beta|$

$\downarrow$  NON È  $V^+$  MA È  $V^* \cdot V_N \cdot V^*$

· GRAMMATICA DI TIPO 0:

SONO TUTTE QUELLE GRAMMATICHE CARATTERIZZATE DA PRODUZIONI DEL TIPO

$$\alpha \rightarrow \beta$$

CON  $\alpha \in V^* \cdot V_N \cdot V^*$  E  $\beta \in V^*$ .

\* PER LE GRAMMATICHE DI TIPO 0 SI AMMETTE LA E-PRODUZIONE E SI POSSONO AVERE PRODUZIONI CON PARTI DESTRE PIÙ PICCOLE DELLA PARTE SINISTRA

PER LE GRAMMATICHE DI TIPO 1 NON SI AMMETTE LA E-PRODUZIONE E SONO CONTEXT-SENSITIVE.

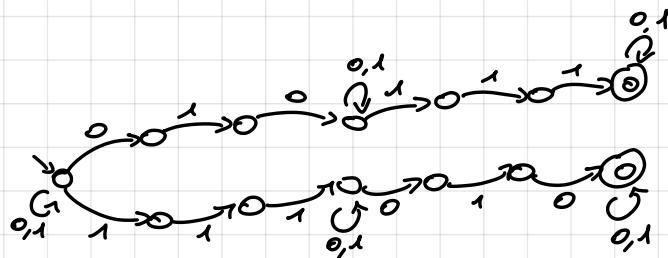
~~PER LE GRAMMATICHE DI TIPO 2 NON SI AMMETTE LA E-PRODUZIONE E SONO CONTEXT-FREE, SONO RICONOSCUTE DA ASF A PILA.~~

PER LE GRAMMATICHE DI TIPO 3 NON SI AMMETTE LA E-PRODUZIONE. LE POSSONO TUTTORA RAPPRESENTARE IN FORMA ALGEBRICA DA ER E SONO RICONOSCUTE DA ASF DET. E NON DET. EQUIVALENTEMENTE.

\*<sub>1</sub> SONO SEMI DECIDIBILI !!

\*<sub>2</sub> POSSIAMO FARE PARSING PREDITTIVO PER I DET.

G) Posso avere in mezzo qualsiasi cosa, non ho strisce fatte solo da 0 1 0 e 1 1



ii)

$$L = \{0^n 1^m \mid m \geq 0, n \geq 0, m < n\}$$

$S \rightarrow MN \mid_1$

$M \rightarrow_0 M_1 \mid_{01} \mid_1$

$$N \rightarrow \epsilon N \Big|_1$$

$$S \rightarrow OS_1 | T$$

$$T \rightarrow {}_1 T \mid {}_1$$

1)

$$S \rightarrow aSb \mid aAb$$

$$A \rightarrow aaAbb | ab$$

$$1) S \Rightarrow aAb \Rightarrow aaaAbbbb \Rightarrow aaaa bbbbb$$

$$2) L = \{ a^n b^n \mid n \geq 2 \}$$

3)  $\sin(-1, 1)$

1 - 1

IN EATTI PER OTE NERI. ad 4 bb mi BASTA PROUD'HEM & VOLTE & SB OPPURE 2 A b

2)

LA CLASSE P È L'INSIEME DI TUTTI I LINGUAGGI DECIDIBILI IN TEMPO POLINOMIALE DA UNA MT. DET.

LA CLASSE NP È L'INSIEME DI TUTTI I LINGUAGGI  $L$  PER LI SR  $x \in L$  ALLORA ESISTE IL CERTIFICATO  $C(x)$  CON LE SEGUENTI CARATTERISTICHE:

- $|C(x)|$  È POLINOMIALE RISPETTO A  $|x|$
- UNA MT DET. CON IN INPUT  $x$  E  $C(x)$  È IN grado DI DECIDERE SE  $x \in L$  IN TEMPO POLINOMIALE RISPETTO  $|x|$  E  $|C(x)|$

3) VOGLIO DIMOSTRARE CHE SE  $A \leq_p B$  E  $B \leq_p C$  ALLORA  $A \leq_p C$ .

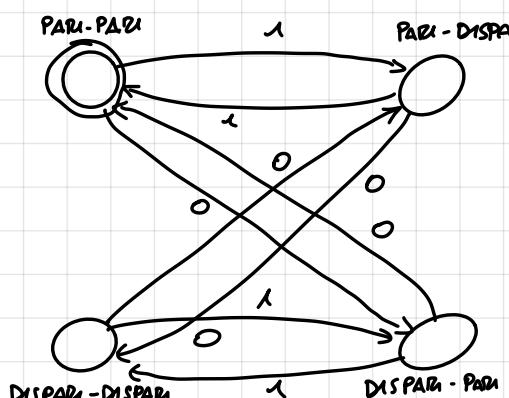
SE  $A \leq_p B$  ALLORA ESISTE  $f_{AB}$  FUNZIONE DI RIDUZIONE TALE CHE SE  $A$  HA ISTANZA SI ALLORA  $f_{AB}$  HA UNA ISTANZA SI DI  $B$  E LO STESSO PER IL NO.

SE  $B \leq_p C$  ALLORA ESISTE  $f_{BC}$  CON LE STESE CARATTERISTICHE.

ABBIAMO ALLORA CHE  $f_{AC} = f_{BC} \circ f_{AB}$ , QUINDI ESISTE LA FUNZIONE DI RIDUZIONE E QUINDI  $A$  È RIDUCIBILE A  $C$ .

4) IGNORANDO COSTANTI MOLTIPLICATIVE ED ADDITIVE SI VA A SEMPLIFICARE L'ANALISI SULLA COMPLESSITÀ CONSIDERANDO CHE GIÀ DI PER SE LE COSTANTI SONO UN'APPROSSIMAZIONE DEL COSTO DELLE SINGOLE ISTRUZIONI. QUESTO, PERO, PORTA AD AVERE CHE, NEL CASO DI COSTANTI ELEVATE, 2 ALGORITMI CON STESSO COSTO ASINTOTICO, PER IL CASO DI DIMENSIONI DELL'INPUT PICCOLI, RICHIEDERANNO TEMPI MOLTO DIVERSI. CON L'ANALISI ASINTOTICA QUESTI PARTICOLARI SI VANNO A PERDERE.

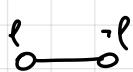
7)



1) COME RIDUZIONE SCIEGO  $3\text{-SAT} \leq_p VC$

SCRIVO 3-SAT COME UNA ISTANZA DI VC. PER FARLO CIÒ DEFINISCO UN GRAFO CARATTERIZZATO DA:

UN  $K_2$  PER OGNI LETTERALE  $\ell$

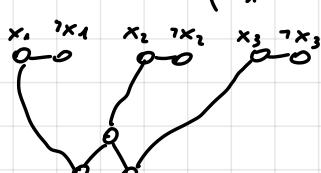


UN  $K_3$  PER OGNI CLAUSOLA



AD OGNI NODO DELLA CLAUSOLA ASSOCIO POI UN LETTERALE DISEGNANDO UN ARCO DA QUEL NODO AD UNO DEI NODI DEL  $K_2$ .

AD ESEMPIO DATO  $(x_1 \vee x_2 \vee \neg x_3)$  HO



ABBIAMO ALLORA CHE LA FORMULA CNF AMMETTE SOLUZIONE SE E SOLO SE ESISTE UN VERTEX COVER DI CARDINALITÀ  $l+2m$  DEL GRAFO PRESENTATO, DORRE H È IL NUMERO DI LETTERALE ED M È IL NUMERO DI CLAUSOLE.

QUINDI ABBIAMO UNA RIDUZIONE  $3\text{-CNF} \leq_p VC$  PERCHÉ HO UNA 3-CNF UNA ISTANZA SI QUANDO VC UNA ISTANZA SI, VALE LO STESSO PER IL NO.

FORMULA AMMISSIONE  $\Rightarrow \exists$  VC DI CARDINALITÀ  $l+2m$

PER OTTENERE UN VERTEX-COVER A PARTIRE DA UNA COMBINAZIONE CHE SODDISFA LA FORMULA MI BASTA PRENDERE IN CONSIDERAZIONE  $\ell$  SE LA FORMULA È SODDISFAVA CON  $\ell = \text{TRUE}$  E  $\neg \ell$  SE LA FORMULA È SODDISFAVA CON  $\ell = \text{FALSE}$ . QUESTO VIENE FATTO PER OGNI LETTERALE  $\ell$ , COSÌ COPRO L'ARCO DI OGNI  $K_2$  ED GLI ARCI CHE PARTONO DA UN NODO DI OGNI  $K_2$  AI  $K_3$ .

PER I RESTANTI ARCI BASTA PRENDERE 2 NODI DI  $K_3$  IN MODO TAK CHE COPRANO I 3 ARCI DEL  $K_3$  ED EVENTUALI 3 ARCI RIMANENTI NON COPERTI CHE Vanno DA  $K_2$  A  $K_3$ .

Abbiamo quindi una copertura di cardinalità  $h+2m$ .

$\exists VC$  DI CARDINALITÀ  $h+2m \Rightarrow$  LA FORMA 3-CNF È SODDISFAZIONE

UN VC DI CARDINALITÀ  $h+2m$  È CARATTERIZZATO PROPRIO COME DENTRO PRIMA NECESSARIAMENTE. SE  $R \in VC$  ALLORA  $\ell = \text{TRUE}$ , SE  $\neg R \in VC$  ALLORA  $\ell = \text{FALSE}$ .

Abbiamo quindi dimostrato la riducibilità da 3-CNF a VC.

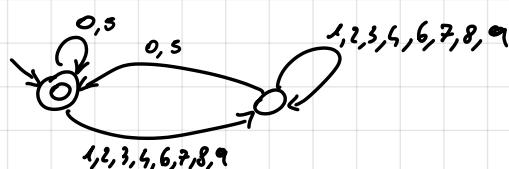
1)  $L = \{2^n 3^{n+1}, n \geq 0\}$  DEFINIRE UNA GRAMMATICA CHE GENERA  $L$

$S \rightarrow S T_3$

$$T \Rightarrow 2T_3 \mid 3$$

NON ESISTE UN ASF PERCHÉ NON ABBIANO MODO DI CONTARE QUANTI S E QUANTI 3 ABBIANO PER OGNI N AVENDO UN NUMERO FINITO DI STATI

2)



3) AMMETTIAMO CHE IL NASTRO SIA ILLIMITATO SOLO A DESTRA. ALLORA OGNI VOLTA CHE SCRIVO UN CARATTERE IN UNA CASELLA BLANK A SINISTRA DELLA SEQUENZA DI CARATTERI NON BLANK SPOSTO DI 1 CASELLA A DESTRA TUTTI I CARATTERI DELLA SEQUENZA.  
STESO RAGIONAMENTO PUÒ ESSERE FATTO SE LA PARTE UNITATA È A DESTRA.

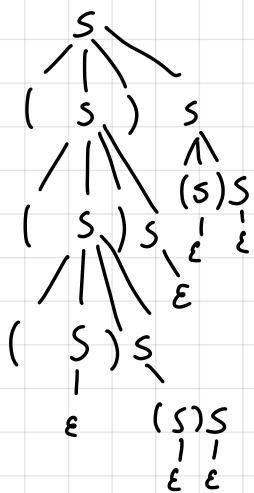
4)

$$S \rightarrow (S)S \mid \epsilon$$

É LL(1) INFATTI

$$\text{SELECT}(S \rightarrow (s)s) = \{ (\ ) \} \cap \text{SELECT}(S \rightarrow \epsilon) = \{ \ ), \$ \} = \emptyset$$

STRUNGA (((()()())()



7)

$$T(h) = T(h-1) + \lambda$$

$$T_{(h-1)} = T_{(h-2)} + 1$$

•

SBAGLIATO:  $\frac{(n-1)n}{2}$  PERCHÉ È SOMMA DEI PRIMI  $n-1$  NUMERI NON  $n$

QUINDI  $T(n) = \frac{(n+1)n}{2} = \Theta(n^2)$  CIOÈ PARI ALLA SOMMA DEI PRIMI  $n$  NUMERI  
 → È  $n-1$  NUMERI

$$T(10) = \frac{110}{2} = 55$$

## QUESTO 1

1.1 Dopo a no 2b

$$S \rightarrow bS \mid aA \mid b$$

$$A \rightarrow bB$$

$$B \rightarrow bS$$

1.2

IL LINGUAGGIO È DI TIPO 3 AVENDO TROVATO UNA GRAMMATICA DI TIPO 3 CHE LO GENERA

2. OPERATORI  $\wedge, \vee, \neg \rightarrow$  LINGUAGGIO DELLE FORMULE BOOLEANE CORrette  $\top \rightarrow$  VARIABILE

$$S \rightarrow S \vee S \mid A$$

$$A \rightarrow A \wedge A \mid \top$$

$$T \rightarrow \top \mid \neg \top \mid (S)$$

→ SERVE SE VOGLIO CHE UN'ESPRESSIONE SIA TRA PARENTESI. HA MASSIMA PRIORITÀ PERCHÉ PRIMA DISOLVO LE ESPRESSIONI TRA PARENTESI E Poi ciò che c'è fuori

3) UNA GRAMMATICA SI DICE AMBIGUA QUANDO AMMETTE PIÙ DI 1 ALBERO DI DERIVAZIONE PER ALMENO 1 STRINGA DEL LINGUAGGIO.

UN ESEMPIO DI GRAMMATICA AMBIGUA È

$$S \rightarrow dS \mid aA \mid a$$

$$A \rightarrow aS \mid a$$

PER GENERARE LA STRINGA dd ABBIAMO I SEGUENTI ALBERI DI DERIVAZIONE:



LA GRAMMATICA PRESENTATA È QUINDI AMBIGUA.

UNA GRAMMATICA USATA PER GENERARE UN LINGUAGGIO DI PROGRAMMAZIONE NON DEVE ESSERE AMBIGUA PERCHÉ PORTEREBBE A MODIFICARE LA SEMANTICA DI UN PROGRAMMA E RENDENDO, QUINDI, IL PROGRAMMA ERRATO.

4) UNA GRAMMATICA CF SI DICE LL(1) SE, PER OGNI NON TERMINALE DELLA GRAMMATICA, NO Oltre A COPPIE DI PRODUZIONI SU TALE NON TERMINALE HANNO INTERSEZIONE DEI SELECT NULLA. \*

QUANDO LA GRAMMATICA È LL(1) HO LA POSSIBILITÀ DI SVOLGERE UN PARSING PREDITTIVO LEGGENDO I CARATTERI SUCCESSIVI DELL'INPUT. Attraverso la lettura di QUESTO CARATTERE SO QUALE PRODUZIONE DI UN NON TERMINALE APPLICARE STAMPA FADE ERRORE GRAFICHE ALL'INTERSEZIONE NULLA FRA I SELECT.

NEL CASO IN CUI IL CARATTERE CHE LEGGO NON FA PARTE DI ALCUN SELECT DELLE PRODUZIONI DISPONIBILI SUL NON TERMINALE ORA CONSIDERATO ALLORA POSSO DIRI DIRETTAMENTE CHE LA STRINGA NON APPARTIENE AL LINGUAGGIO.

\* CONVIENE SCRIVERE PIÙ FORMALE:

DATO UN NON TERMINALE A:  $A \rightarrow a \in A \Rightarrow B$  ALLORA G È LL(1) SE $\text{SELECT}(A \rightarrow a) \cap \text{SELECT}(A \Rightarrow B) = \emptyset$  E QUESTO DEVE VALERE PER OGNI NON TERMINALE DI G

5)

LA CLASSE P È L'INSIEME DI LINGUAGGI  $L$  PER CUI UNA MT DET. DECIDE SE  $x \in L$  IN TEMPO POLINOMIALE.

LA CLASSE NP È L'INSIEME DI LINGUAGGI  $L$  PER CUI SE  $x \in L$  ALLORA  $C(x)$ , DETTO CERTIFICATO, IL QUALE HA LE SEGUENTI CARATTERISTICHE:

- $|C(x)|$  È POLINOMIALE RISPETTO  $|x|$
- UNA MT DET CON IN INPUT  $C(x)$  ED  $x$  È IN GRADO DI DECIDERE SE  $x \in L$  IN UN TEMPO POLINOMIALE RISPETTO  $|C(x)|$  E  $|x|$

NEL PROBLEMA P vs NP CI SI CHIEDE SE  $P = NP$  OPPURE  $P \neq NP$ .

PER DEMONSTRARE CHE  $P \neq NP$  BISOGNA TROVARE UN PROBLEMA CHE SI TROVA STRETTAMENTE IN NP, PER FARLO CIÒ SERVE TROVARE ALMENO 1 PROBLEMA CON LOWER BOUND PIÙ CHE POLINOMIALE PER UNA MACCHINA DET. MA PER ORA NON È STATO TROVATO.

PER DEMONSTRARE CHE  $P = NP$  BISOGNA DEMONSTRARE CHE UN PROBLEMA NP-COMPLETO HA UPPER BOUND POLINOMIALE PER UNA MACCHINA DET. E CIÒ È TROVARE UN ALGORITMO CHE RISOLVE IL PROBLEMA IN TEMPO POLINOMIALE.

6)

## 6.1

3-COLORABILITÀ È UN PROBLEMA NP-COMPLETO. QUESTO SIGNIFICA CHE, TROVANDO UN UPPER-BOUND  $O(n^6)$ , HANNO TROVATO UN UPPER-BOUND POLINOMIALE. CIÒ VUOL DIRE CHE  $P = NP$  INFATTI ESSENDO NP-COMPLETO POSSO RIDURRE POLINOMIALMENTE TUTTI I PROBLEMI A 3-COL.

6.2) VERTEX-COVER È UN PROBLEMA NP-COMPLETO, QUINDI SE TROVO CHE IL LOWER BOUND È NON POLINOMIALE PER UNA MACCHINA DET. DEMOSTRO CHE  $P \neq NP$ , INFATTI HO MOSTRATO CHE VC È NP STRETTAMENTE.

## 6.3

3-SAT È NP-COMPLETO QUINDI SAPPIANO GIÀ CHE L'UPPER BOUND È ESPONENZIALE PER UNA MACCHINA DET.

## ▷ ESTATE FEBBRAIO 2020

1) DATE 2 FUNZIONI  $f(x)$  E  $g(x)$  SI DICE CHE  $f(x) \in O(g(x))$  SE  $f(x)$  CRESCHE AL MASSIMO QUANTO  $g(x)$  PER  $x \rightarrow +\infty$ .

UN PROBLEMA P È COMPUTAZIONALMENTE FACILE SE APPARTIENE ALLA CLASSE P, È CIÒ DECIDIBILE DA UNA MT DET. IN TEMPO POLINOMIALE.

2) PER MOSTRARE CHE VC È NP BASTA CONSIDERARE UN CERTIFICATO  $C(x)$  DOVE  $x$  È UN POSSIBILE VERTEX COVER DI UN GRAFO  $G$ . ABBIANO CHE  $|C(x)|$  È POLINOMIALE RISPETTO  $|x|$  INFATTI HO SEMPLICEMENTE LA LISTA DEI NODI DEL VC  $x$ .

UNA MT DET. PUÒ VERIFICARE SE  $x$  È UN VC DI CARDINALITÀ  $K$  DI  $G$  IN TEMPO POLINOMIALE, INFATTI BASTA VERIFICARE SE TUTTI GLI ARCI DI  $G$  HANNO ALMENO 1 NODO NEL VC. QUESTO PUÒ ESSERE FATTO SCANDENDO LA LISTA DEGLI ARCI, CIÒ HA COSTO POLINOMIALE IN  $|C(x)|$  E QUINDI VC È NP.

IL PROBLEMA VERTEX COVER IN FORMA DECISIONALE PRENDE IN INPUT UN GRAFO  $G = \langle V, E \rangle$  ED UN INTERO  $K$ , IN OUTPUT SI HA SI SI ESISTE UN INSERTE  $V'$  PER CUI SE  $e = \langle v, u \rangle \in E$  ALLORA  $v \in V'$  OR  $u \in V'$  ALTRIMENTI IN OUTPUT SI HA RISPOSTA NEGATIVA.

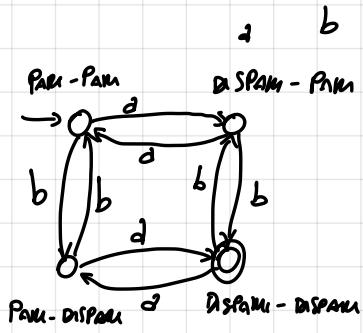
4) UNA MT UNIVERSALE PRENDE IN INPUT LA DESCRIZIONE DI UNA MT E L'INPUT DELLA MT DA SIMULARE. UNA MT UNIVERSALE, QUINDI, SIMULA LA MT DESCRITA PERÒ TERMINA LA COMPUTAZIONE SE LA MACCHINA SIMULATA TERMINA, ALTRIMENTI NON TERMINA. IL RISULTATO DELLA COMPUTAZIONE È IL RISULTATO DELLA MACCHINA SIMULATA.

LA MT UNIVERSALE FORMA ANCHE IL CONCETTO DI INTERPRETE.

5) [UN PROBLEMA SI DICE INDECIDIBILE SE, PER ALMENO UN INPUT DEL PROBLEMA, LA COMPUTAZIONE DELLA MT CHE RISOLVE QUESTO PROBLEMA NON TERMINA.]  
→ SBAGLIATO, LA RISPOSTA GIUSTA È:

• UN PROBLEMA SI DICE INDECIDIBILE SE NON ESISTE alcun ALGORITMO CHE LO RISOLVE

7) STRUTTURA CON NUM. DISPARI DI 2 E NUMERO DISPARI DI b



8)

$$\begin{aligned} S &\rightarrow S_d \\ S &\rightarrow A b \\ S &\rightarrow A \\ A &\rightarrow A b \\ A &\rightarrow b \end{aligned}$$

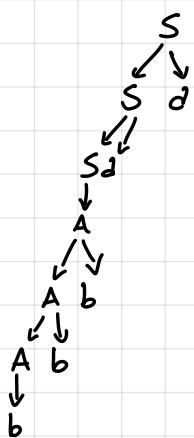
STRINGA bbbdd

LA GRAMMATICA È DI TIPO 2. È AMBIGUA PERCHÉ PER PRODURRE bbbdd POSSO APPLICARE ANCHE LE SEGUENTI DERIVAZIONI:

$S \Rightarrow S_d \Rightarrow Sdd \Rightarrow A dd \Rightarrow Ab dd \Rightarrow Abbdd \Rightarrow bbbdd$  OPPURE  $S \Rightarrow S_d \Rightarrow Sdd \Rightarrow A b dd \Rightarrow Abbbaa \Rightarrow bbbdd$

ATTENZIONE:

QUI SI CHIEDEVA UN ALBERO DI DERIVAZIONE, ESSO È IL SEGUENTE



9)  $S \rightarrow S+S \mid S-S \mid T$   
 $T \rightarrow id \mid (S)$

3) PER  $n=4$  NO 7 CHIAMATE RICORSIVE

STUDIO ORA LA COMPLESSITÀ IN FUNZIONE DI n

IMPORTANTE

$$\left\{ \begin{array}{l} T(n) = T(n-1) + T(n-2) + 1 \leq 2T(n-1) + 1 = 2[T(n-2) + T(n-3) + 1] + 1 \leq 2[2T(n-2) + 1] + 1 = 2^n T(1) + \sum_{i=0}^{n-1} i = O(2^n) \\ T(1) = 1 \quad T(0) = 1 \end{array} \right.$$

## ▷ PDF DOMANDE Piazza

5.

IL COSTO È DATO DA

$$T(n) = 1 + 2 + 3 + 4 + \dots + n - 1 = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = \Theta(n^2)$$

SE  $n = 10$

$$T(10) = \frac{10(9)}{2} = 45$$



6.  $T(n) = \frac{n}{2} \cdot \log_2 n = \Theta(n \log n)$

PER  $n=10$  NO

$$T(10) = \frac{10}{2} \log_2 10 = 6 \log_2 10$$

→ FACCIO UN SOMME PER OGNI COLONNA

7. SONO NECESSARIE  ~~$m \cdot n$~~  OPERAZIONI

## COMPUTABILITÀ

1. LA TESE DI CHURCH-TURING È CHE TUTTO CIÒ CHE È CALCOLABILE MEDIANTE UN ALGORITMO È CALCOLABILE CON UNA MACchina DI TURING.

VIENE DETTA TESE PERCHÉ NON È STATA DEMONSTRATA, BISOGNEREBBE PROVARLO PER OGNI ALGORITMO. NON È DEMONSTRABILE PERCHÉ POTREBBE ESSERE CHE QUESTA TESE NON VADA PER ALGORITMI CHE NON CONOSCIAMO.

3. B E C DECIDIBILI

• B U C

PER RICONOSCERE B U C POSSO APPLICARE PRIMA A<sub>1</sub> E Poi A<sub>2</sub> ALLA STRINGA IN INPUT CONSIDERANDO CHE DA A<sub>1</sub> CHE A<sub>2</sub> TERMINANO SUERO. SE UNO DEI 2 ALGORITMI HA SUCCESSO ALLORA LA STRINGA APPARTIENE A B U C

• B ∩ C

PER RICONOSCERE B ∩ C POSSO APPLICARE SIA A<sub>1</sub> CHE A<sub>2</sub>, SE TUTTI E 2 HANNO SUCCESSO ALLORA LA STRINGA APPARTIENE A B ∩ C.

• B/C

PER RICONOSCERE B/C POSSO APPLICARE A<sub>1</sub> CHE RICONOSCE B E D APPLICARE A<sub>2</sub>: SE A<sub>1</sub> DA ESITO POSITIVO ED A<sub>2</sub> FALLISCE ALLORA LA STRINGA APPARTIENE A B, ALTRIMENTI NO.

• B\*

PER RICONOSCERE B\* POSSO APPLICARE TANTE VOLTE A<sub>1</sub> QUANTE SONO LE SOTOSTRINGS CHE APPARTENGONO A B DELLA STRINGA IN INPUT.

→ SI PUÒ FARE BRUTE-FORCE: APPLICO A<sub>1</sub> E RICONOSCO UNA SOTOSTRINGA, A QUESTO PUNTO PASSO ALLA SOTOSTRINGA SUCCESSIVA E COSÌ VIA. SE TUTTE LE GUARDA DANNANO ESITO POSITIVO LA STRINGA APPARTIENE AL LINGUAGGIO, ALTRIMENTI, SE UNA GUARDA FALSE SI TORNA INDIRITTO ED ALUNCO LA SOTOSTRINGA PRECEDENTE CERCANDO DI RICONOSCERLA UNA PIÙ LUNGA E COSÌ VIA.

## IMPORTANTE

4. PER MOSTRARE CHE L'INSIEME È DECIDIBILE MI BASTA MOSTRARE UN MODO PER VERIFICARE SE UN AUTOMA APPARTIENE ALL'INSIEME.

SIA X UN ASF. PER DECIDERE SE X APPARTIENE AL LINGUAGGIO COSTRUISCO L'AUTOMA DETERMINISTICO SEGUENDO LA CODIFICA SCRITTA IN X.

A QUESTO PUNTO VERIFICO PER OGNI STATO FINALE q SE ESISTE UN CAMMINO DALLO STATO INIZIALE A q. SE TALE CAMMINO ESISTE PER ALMENO UNO STATO FINALE ALLORA L'AUTOMA ACCETTA ALMENO UNA STRINGA. PER FARLO CIÒ È SUFFICIENTE VERIFICARE SE CI SONO CAMMINI DI LUNGHEZZA AL PIÙ |Q| - 1 PERCHÉ SE VOI CAMMINI PIÙ LUNGI SIGNIFICA SEMPREMENTE CHE ENTRO IN UN CICLO.

## GRAMMATICA, ER ED ASF

→ IN QUESTO CASO PERÒ SI PARLA DI PROGRAMMA

1. L'ANALISI LESSICALE DI UNA STRINGA CONSISTE NEL TRADURRE UNA STRINGA DI CARATTERI IN UNA STRINGA DI TOKEN E CIOÈ IN UNA SEQUENZA DI TERMINI DELLA GRAMMATICA. SE L'ANALISI LESSICALE TROVA 1 O PIÙ CARATTERI CHE NON RAPPRESENTANO UN TOKEN FALLISCE, ALTRIMENTI FORNISCE IN OUTPUT LA STRINGA DI TOKEN ASSOCIATA ALLA STRINGA DI CARATTERI IN INPUT.

L'ANALISI SINTATTICA DI UNA STRINGA PRENDE IN INPUT L'OUTPUT DELL'ANALISI LESSICALE E DA IN OUTPUT L'ALBERO DI DERIVAZIONE SE LA STRINGA DI TOKEN È GENERATO DALLA GRAMMATICA, ALTRIMENTI FACCERÀ DA IN OUTPUT UNA DIAGNOSTICA SU QUALE CARATTERE È ERRORE.

### 3. GRAMMATICHE DI TIPO 3

SONO GRAMMATICHE REGOLARI E SONO CARATTERIZZATE DALLE PRODUZIONI

$$A \Rightarrow a, A \Rightarrow B$$

$$A, B \in V_N, a \in V_T$$

### GRAMMATICHE DI TIPO 2

SONO GRAMMATICHE CONTEXT-FREE E SONO CARATTERIZZATE DA PRODUZIONI DEL TIPO

$$A \Rightarrow B$$

$$A \in V_N, B \in V^+ \quad \text{E} \quad |B| \geq |A|$$

### GRAMMATICHE DI TIPO 1

SONO GRAMMATICHE CONTEXT-SENSITIVE ED HANNO PRODUZIONI DEL TIPO

$$\lambda A B \Rightarrow B$$

$$\lambda, \beta \in V^*, A \in V_N, B \in V^+ \quad \text{E} \quad |\lambda A \beta| \leq |B|$$

### GRAMMATICHE DI TIPO 0

SONO GRAMMATICHE CHE HANNO PRODUZIONI DEL TIPO

$$\lambda A B \Rightarrow B$$

$$\text{con } A \in V_N, \lambda, \beta \in V^*, B \in V^*$$

**IMPORTANTE**

LE GRAMMATICHE DI TIPO 0 GENERANO LINGUAGGI SEMI-DECIDIBILI.

LE GRAMMATICHE DI TIPO 1 GENERANO LINGUAGGI DECIDIBILI

LE GRAMMATICHE DI TIPO 2 GENERANO LINGUAGGI RICONOSCIUTI DA AUTOMI A PILA, IN QUESTO CASO IL NON DET. AUMENTA IL POTERE RICONOSCITIVO DELL'AUTOMA.

LE GRAMMATICHE DI TIPO 3 GENERANO LINGUAGGI RICONOSCIUTI DA AUTOMI A STATI FINITI, IL NON DET. NON AGGIUNGE POTERE ESPRESSIVO AGLI AUTOMI

3. DATA UNA RICORSIONE SINISTRA PER RIACQUERIRE SCRIVO UN NUOVO NON TERMINALE CON UNA SOLA PRODUZIONE CHE GENERA I TERMINAL CHE CUIANO LA RICORSIONE SINISTRA ED UN NON TERMINALE CHE HA COME PRODUZIONI UNA RICORSIONE DESTRA CHE MI GENERA LA SEQUENZA DI TERMINAL ED UNA E-PRODUZIONE.

4.

$\overset{\text{E' PIÙ CONODO COSÌ}}{T}$

$$S \rightarrow T + S \mid T - S \mid T$$
$$T \rightarrow id \mid (S)$$

5. NON PUÒ ESSERE GENERATO DA UNA GRAMMATICA DI TIPO 3 PERCHÉ NON VALGONO IL PUMPING LEMMA, QUESTO PERCHÉ IL LINGUAGGIO PUÒ GENERARE PIÙ PARENTESE TONDE ALL'INIZIO QUINDI, SE VOGLIO APPLICARE IL PUMPING, DEVO ANCHE RICONOSCERE LO STESSO NUMERO DI PARENTESE TONDE CHIUSE E QUESTO IMPLICA CHE L'AUTOMA ABbia ALMENO 2P STATI SE PIÙ IL NUMERO DI PARENTESE APerte. SE HO UNA STRINGA CON P+1 PARENTESE APERTA NON POSSO FAR PUMPING, NON PUOSSERMI AD AVERE LO STESSO NUMERO DI PARENTESE APERTE E CHIUSE.

UNA GRAMMATICA È LL(1) SE PER OGNI PRODUZIONE DEL TIPO

$$\alpha \rightarrow \beta_1 \mid \beta_2$$

SI HA CHE

$$\text{SELECT}(\alpha \Rightarrow \beta_1) \cap \text{SELECT}(\alpha \Rightarrow \beta_2) = \emptyset$$

SONO MOLTO ADATTE PER DESCRIVERE LINGUAGGI DI PROGRAMMAZIONE. PERCHÉ POSSO APPLICARE UN PARSING PREDITTIVO EFFICIENTE.

6. UNA GRAMMATICA LR(0) È UNA GRAMMATICA CHE NON HA CONFLITTI REDUCE-REDUCE E SHIFT-REDUCE, PERCÒ NON HA AMBIGUITÀ NELLA TAVOLA ACTION IN ALLEN STATO.

## IMPORTANTE

7. ALGORITMO DI ANALISI SINTATTICA LR(0):

I PARSER LR(0) UTILIZZANO UNA PILA LA QUALE CONTIENE GLI STATI CHE VENGONO USATI, INSIEME ALLA LETTURA DEL CARATTERE DELL'INPUT, PER DECIDERE LE AZIONI DA SVOLGERE INTERPRELLANDO LE SEGUENTI TAVOLE

- ACTION [s, a] RIGHE → STATI COLONNE → TERMINALI
- GoTo [s, X] RIGHE → STATI COLONNE → NON TERMINALI

### Action [s, a]

- DESCRIVE QUALE AZIONE ESEGUIRE QUANDO LO STATO AFFIORANTE IN PILA È s ED IL PROSSIMO TOKEN IN INPUT È IL TERMINALE a

AZIONI POSSIBILI:

- SHIFT s: INSERISCO NELLA PILA LO STATO s (push)
- REDUCE: FACCIO POP DEGLI STATI CHE RAPPRESENTANO L'HANDLE DELLA PRODUZIONE DI CUI DEVO FARRE REDUCE. AVRÒ ORA UNO STATO AFFIORANTE, ATTRAVERSO QUESTO STATO INSERIRE AL NOVE TERMINALE DELLA PRODUZIONE CON CUI STO FACENDO LO REDUCE. CONDOTTO LA TAVOLA GOTO E CAPISCO DA QUALE STATO FARRE IL PUSH.
- ACCEPT: TERMINA CON SUCCESSO
- REPORT ERROR: SE NON È POSSIBILE PROCEDERE

### GoTo[s, X]

- INDICA QUALE NUOVO STATO PIANTARE IN CIMA ALLA PILA (push) DOPO LA RIDUZIONE DEL NON TERMINALE X, MENTRE LO STATO AFFIORANTE È s.

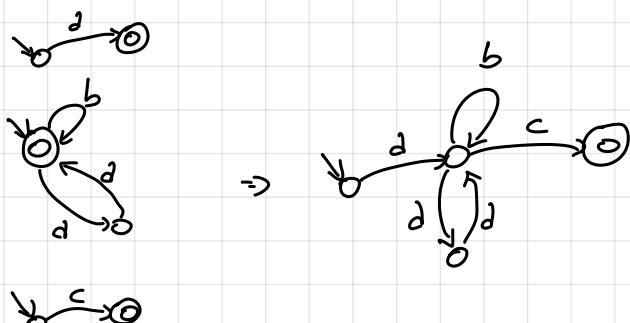
10.

$$\begin{array}{ll} S \rightarrow d b c & \text{IL LINGUAGGIO GENERATO È } \{d^n b^n c^n : n \geq 1\} \\ S \rightarrow d S B c \\ C B \rightarrow B c \\ B B \rightarrow b b \end{array}$$

LA GRAMMATICA È DI TIPO 1. NON È POSSIBILE DETERMINARE UNA GRAMMATICA DI TIPO 3 PERCHÉ NON VALE IL PUMPING LEMMA PER I LR.

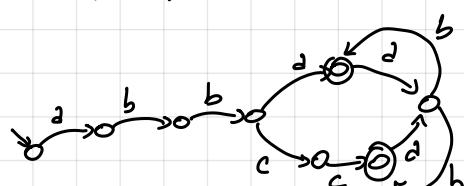
11.

1)  $R_1 = a(b|dd)^*c$



Ti voglio tanto bene   
Non smettere mai di credere  
in te stesso e nelle tue potenzialità.  
- Elisa -

2)  $a b b (d|cc)(ab)^*$



## 12. ← IMPORTANTE

SE IL LINGUAGGIO  $L$  RICONOSCIUTO DA UN ASF CON  $n$  STATI NON È VUOTO ESISTE UNA STRUNGA  $x$  IN  $L$  DI LUNGHEZZA  $|x| < n$ .

PER VERIFICARE SE UN LINGUAGGIO È VUOTO BASTA PROVARE LE STRUNGHE CON LUNGHEZZA MINORE DEL NUMERO DI STATI DELL'AUTOMA. DIAVOLO QUINDI UN INPUT L'INSIEME DI STRUNGHE CON QUESTA CARATTERISTICA IL QUALE È SICURAMENTE FINITO, SE NON RAGGIUNGO LO STATO FINALE PER ALCUNA STRUNGA ALLORA IL LINGUAGGIO È VUOTO.

## 13. ← IMPORTANTE

PER AVERE UN LINGUAGGIO INFINTO DEVO AVERE NECESSARIAMENTE UN CICLO. PER VERIFICARNE L'ESISTENZA DO IN INPUT ALL'AUTOMA L'INSIEME DI STRUNGHE CON LUNGHEZZA  $n < |x| < 2n$ , INNEHE FINITO. SE L'AUTOMA ACCETTA ALMENO UNA STRUNGA L'INSIEME È INFINTO PERCHÉ ESISTE UN CICLO, ALTRIMENTI IL LINGUAGGIO È FINITO.

15.

1)  $(0+1)^*(101+010)^*(0+1)^*(101+010)^*$

2)  $(0+1)^*(00(0+1)^*11(0+1)^* + 11(0+1)^*00(0+1)^*)$

3)  $(0+10)^*00(0+10)^*$

4)  $00(0+1)^*11$

5)  $(xy+z+yz)^+$

6)  $(x+z)^*y \left[ (z+x)^*y (z+x)^*y \right]^*$

### GRAMMATICHE LIBERE DAL CONTESTO

1.

I NON TERMINALI NON RAGGIUNGIBILI A PARTIRE DALL'ASIOMA SONO: D, F

PER ELIMINARE LE PRODUZIONI INUTILI BASTA ELIMINARE QUELLE LEGATE A D ED F.

2. VEDIAMO CHE B NON DEFINISCE ALMENO UNA STRUNGA PERCHÉ SI ENTRA IN UN CICLO INFINTO.

SENTA LA B QUINDI OTTENIAMO

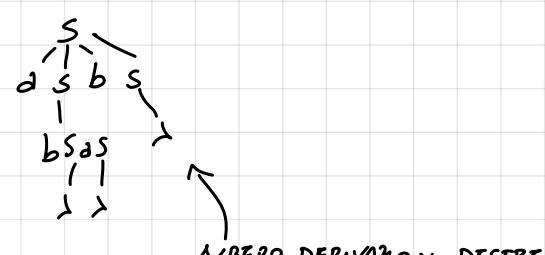
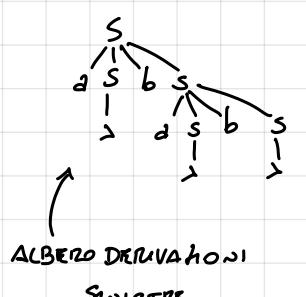
$$\begin{aligned} S &\rightarrow AC \\ A &\rightarrow dAb \mid ab \\ C &\rightarrow c \end{aligned}$$

INFATTI DEVO ELIMINARE TUTTE LE PRODUZIONI CHE COINVOLGONO LA B.

5.

$$S \rightarrow aSbS \mid bSas \mid \lambda$$

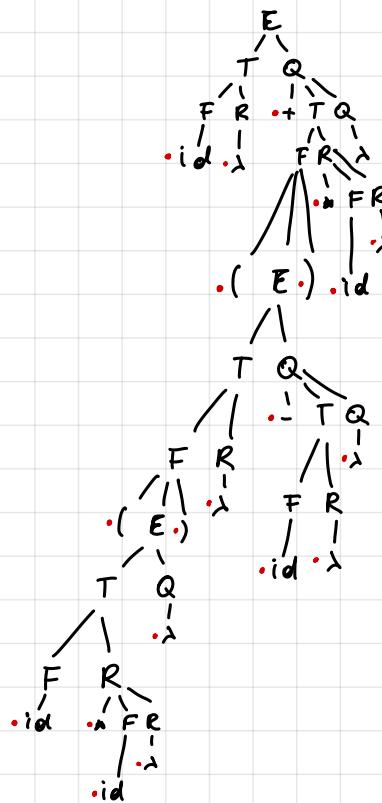
SE LA GRAMMATICA È AMBIGUA, PER  $aabb$  ESISTONO I SEGUENTI ALBERI DI DERIVAZIONE:



6)

$$\begin{aligned} E &\rightarrow TQ \\ Q &\rightarrow +TQ \mid -TQ \mid \lambda \\ T &\rightarrow FR \\ R &\rightarrow ^*FR \mid /FR \mid \lambda \\ F &\rightarrow (E) \mid id \end{aligned}$$

d)  $id + ((id \cdot id) - id) \cdot id$



b) Non può essere calcolata solo con i FIRST perché no E-produzioni, per riconoscerla bisogna dei FOLLOW.

7)

$$S \rightarrow aB \mid aC \mid B \quad S \rightarrow aB \mid B$$

$$B \rightarrow bB \mid d \quad B \rightarrow bB \mid d$$

$$C \rightarrow B$$

Per dimostrare che non è LL(1) calcolo i SELECT delle produzioni

$$\text{SELECT}(S \rightarrow aB) = \{a\} \quad \text{SELECT}(S \rightarrow aC) = \{a\}$$

Vediamo già qui

$$\text{SELECT}(S \rightarrow aB) \cap \text{SELECT}(S \rightarrow aC) \neq \emptyset$$

Quindi la grammatica non è LL(1).

Per renderla LL(1) posso trasformarla nel seguente modo:

La risoluzione standard è aggiungere un non terminale:

$$\begin{array}{l} S \rightarrow aB \mid B \\ \xrightarrow{\hspace{1cm}} \quad S \rightarrow aT \mid B \\ B \rightarrow bB \mid d \\ T \rightarrow B \mid C \\ \quad \quad \quad \rightarrow \text{SI RAGGRUPPANO LE PRODUZIONI CHE HANNO PRESSI COMUNI IN UN NON TERMINALE} \\ B \rightarrow bB \mid d \\ C \rightarrow B \end{array}$$

NP-COMPLETITÀ

1.

La classe P è l'insieme dei linguaggi L per cui una MT det. è in grado di decidere se  $x \in L$  in tempo polinomiale.

La classe NP è l'insieme dei linguaggi L per cui se  $x \in L$  allora esiste  $C(x)$ , detto certificato, con le seguenti caratteristiche:

•  $|C(x)|$  è polinomiale in  $|x|$

• DATA UNA MT det. che prende in input  $x$  e  $C(x)$  è in grado di verificare che  $x \in L$  in un tempo polinomiale rispetto  $|x| + |C(x)|$ .

Si ha che  $P \subseteq NP$ , non si sa se  $P = NP$  oppure  $P \neq NP$

2.

UN PROBLEMA  $P$  SI DICE NP-COMPLETO SE  $P \in NP$  E PER OGNI PROBLEMA IN  $NP$  È SEMPRE POSSIBILE RIDURLO POLINOMICAMENTE A  $P$ .

SI SA CHE  $NP-C \subset NP$  MENTRE PER LA RELAZIONE CON  $P$  NON SI SA DARE CONCLUSIONI.

3. PER STABILIRE CHE  $P \neq NP$  DEVO TROVARE UN LOWER BOUND PIÙ CIECO POLINOMIALE AD UN PROBLEMA IN  $NP$ . HO COSÌ DIMOSTRATO CHE IL PROBLEMA APPARTIENE AD  $NP$  E NON A  $P$ .

4. PER STABILIRE CHE  $P = NP$  DEVO TROVARE UN UPPER BOUND POLINOMIALE AD UN PROBLEMA NP-COMPLETO. IN QUESTO CASO TUTTI I PROBLEMI, ANCORA IN  $NP$ , SONO RESOLVIBILI IN TEMPO POLINOMIALE DA UNA MT DET.

5.

PER DIMOSTRARE CHE UN PROBLEMA  $A$  È NP-COMPLETO DEVO DEMONSTRARE CHE  $A \in NP$  E CHE POSSO RIDURRE TUTTI I PROBLEMI IN  $NP$  AD  $A$  IN TEMPO POLINOMIALE. PER FARLO CIÒ POSSO, AD ESEMPIO, PROVARE A RIDURRE UN PROBLEMA NP-COMPLETO AD  $A$  COSÌ DA EREDITARE LA COMPLETITÀ.

6. IL PROBLEMA DELLA SODDISFACIBILITÀ DI UNA FORMULA LOGICA CONSISTE NELL'VERIFICARE SE UNA FORMULA LOGICA AMMETTE ALMENO UNA COMBINAZIONE DI ASSEGNAZIONI DI VALORI DI VERITÀ ALLE VARIABILI PRESENTI NELLA FORMULA CHE LA RENDE VERA. SE ESISTE ALMENO UNA COMBINAZIONE ALLORA LA FORMULA È SODDISFACIBILE ALTRIMENTI È INSODDISFACIBILE.

UNA FORMULA CNF È UNA FORMULA BOOLEANA CON LE SEGUENTI CARATTERISTICHE:

ABBIANO CLAUSOLE COMPOSTE DA UN NUMERO ARBITRARE DI LITERALI E SONO DI UNA SISTEMATICA FORMA

$$C_i = (l_1 \vee l_2 \vee \dots \vee l_n)$$

E LE CLAUSOLE SONO IN AND TRA DI LORO

$$C_1 \wedge C_2 \wedge \dots \wedge C_m$$

IL PROBLEMA CNF È UN PROBLEMA NP-COMPLETO.

#### • Esercizio parser grammatica

$$S \rightarrow (S)S \mid \epsilon$$

LA GRAMMATICA È LL(1) INFATTI ABBIANO

$$\text{SELECT}(S \rightarrow (S)S) = \{ ( \} \quad \text{E} \quad \text{SELECT}(S \rightarrow \epsilon) = \{ \}, \$ \}$$

L'INTERSEZIONE TRA I DUE INSIEMI È QUINDI NULLA.

```
void start_parse(char[] str){
    if(S(str, 0) >= 0)
        printf("La stringa appartiene al linguaggio");
    else
        printf("La stringa non appartiene al linguaggio");
}
```

```
int S(char[] str, int pos){
    if(str[pos] == "("){
        int par = S(str, pos+1);
        if(par != -1 && str[par+1] == "(" && str[par] == ")")
            return S(str, par+1);
        else if(par != -1 && str[par] == ")")
            return par+1;
    }
    else if(str[pos] == ")"){
        return pos;
    }
    return -1;
}
```

7. UNA FORMULA DNF È CARATTERIZZATA DA CLAUSOLE NELLA SEGUENTE FORMA

$$C_i = (l_1 \wedge l_2 \wedge \dots \wedge l_j)$$

MENTRE LE CLAUSOLE SONO IN OR TRA DI LORO

$$C_1 \vee C_2 \vee \dots \vee C_m$$

IL PROBLEMA DELLA SODDISFACIBILITÀ DI UNA FORMULA DNF È PIÙ SEMPLICE RISPETTO AD UNA FORMULA CNF. BASTA INFATTI TARE IN TODO CUI UNA DELLE CLAUSOLE VIENGA SODDISFATTA, COSÌ HA QUINDI COSTO POLINOMIALE.

8. IL PROBLEMA SAT È RISOLTO CON L'ALGORITMO CHIAMATO DPLL.

## ← IMPORTANTE

TALE ALGORITMO SI BASA SU 3 EURETICHE:

• UNIT-PROPAGATION: PER OGNI CLAUSOLA UNITARIA PRESENTE NELLA FORMULA ASSEGNA IL VALORE DI VERA O FALSO AL LETTERALE IN MODO DA SODDISFARE TALI CLAUSOLE. TAULE ASSEGNAZIONE VIENE EFFETTUATA PER TUTTE LE OCCORRENZE DEL LETTERALE. SE TALE ASSEGNAZIONE SODDISFA LA CLAUSOLA QUESTA VERRÀ ELIMINATA DALLA FORMULA, ALTRIMENTI VIENE ELIMINATO IL LETTERALE DALLA CLAUSOLA.

• PURE-LITERAL ELIMINATION: UN LETTERALE È PURO QUANDO, PER OGNI OCCORRENZA NELLA FORMULA, APPARE SEMPRE NEGATO OPPURE SEMPRE AFFERMATO. PER OGNI LETTERALE PURO SI CANCELLA DALLA FORMULA LA CLAUSOLA IN CUI È PRESENTE.

L'ALGORITMO È QUINDI IL SEGUENTE:

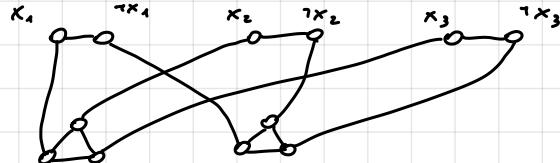
```
DPLL (F) {  
    IF F È VUOTA  
        RETURN TRUE  
    IF F HA ALMENO UNA CLAUSOLA VUOTA  
        RETURN FALSE  
    FOR CLAUSOLA UNITARIA C IN F  
        F := UNIT-PROPAGATION (c, F)  
    FOR LETTERALE PURO l IN F  
        F := PURE-LITERAL-ELIMINATION (l, F)  
    l := CHOOSE-LITERAL  
    RETURN DPLL (F ∪ {l}) OR DPLL (F ∪ {¬l})}
```

9.  $3\text{-CNF} \leq_p VC$

TALE RIDUZIONE CI MOSTRA CHE VC È NP-COMPLETO. PER FARE LA RIDUZIONE DOBBIANO MOSTRARE CHE PER UN'ISTANZA VERA DI 3-CNF SI HA UN'ISTANZA VERA DI VC E VICEVERSA, LO STESSO VALE PER UN'ISTANZA FALSE.

PRENDIAMO IN CONSIDERAZIONE UN GRAFO COMPOSTO DA  $K_2$  E  $K_3$ . ABBIAMO UN  $K_2$  PER OGNI LETTERALE ED UN  $K_3$  PER OGNI CLAUSOLA DELLA FORMULA CONSIDERATA.

AD ESEMPIO SE HO COME FORMULA  $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$  IL GRAFO CORRISPONDENTE È



COLLEGHIAMO QUINDI I NODI DEI  $K_3$  CON QUELLI DEI  $K_2$  IN BASE ALLA FORMULA 3-CNF.

SIANO  $l$  IL NUMERO DEI LETTERALI E  $m$  IL NUMERO DELLE CLAUSOLE.

SI HA CHE ESISTE UN VC DI CARDINALITÀ  $l+2m$  SE E SOLO SE LA FORMULA È SODDISFACIBILE.

$\exists$  VC DI CARDINALITÀ  $l+2m \Rightarrow$  FORMULA SODDISFACIBILE

IL VC SARÀ CARATTERIZZATO DA  $l$  NODI PER OGNI  $K_3$  PIÙ UN NODO PER OGNI  $K_2$  NECESSARIAMENTE. DATO UN LETTERALE  $x$  GLI ASSEGNAVAMO TRUE SE NEL VC HUO IL NODO  $x$ , SE NO IL NODO  $\neg x$  ASSEGNAVAMO FALSE.

FORMULA SODDISFACIBILE  $\Rightarrow$  3 VC DI CARDINALITÀ  $l+2m$

PER EFFETTUARE IL VC BASTA PRENDERE  $l$  NODI DAL  $K_2$  SECONDO L'ASSEGNAZIONE DI VERA CHE SODDISFA LA FORMULA. MENTRE I RESTANTI  $l+m$  NODI VANNO SCRITTI NEI  $K_3$ , VENGONO SLETTI I NODI PER OGNI CLAUSOLA IN MODO CHE VENGANO COPERTI DA ALCUNI RESTANTI, MA QUESTI CHE VENGANO VERSO  $K_2$  SIA QUELLO ALL'INTERNO DEL  $K_3$ , CONSIDERANDO CHE UN ARCO CHE VA VERSO  $K_2$  È STATO GIÀ COPERTO.

10.

A DEVE ESSERE NP-COMPLETO E LA RIDUZIONE DEVE AVERE COSTO POLINOMIALE.

## ► ESEMPIO ANALISI LR(0)

## ← IMPORTANTE

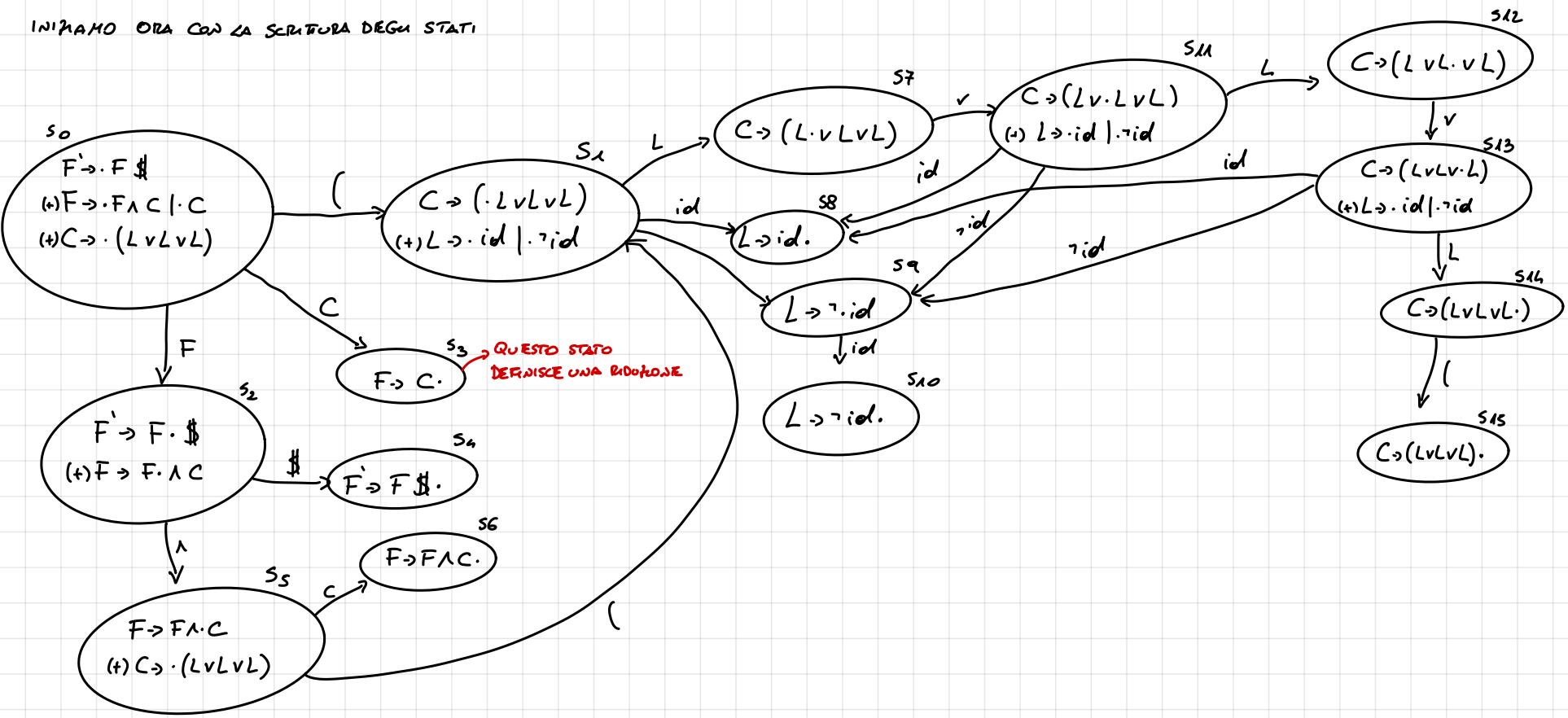
L'ANALISI CONSEGUE NEL PRODURRE IL GOTO - GRAPH

$$\begin{aligned} F &\rightarrow F \wedge C \mid C \\ C &\rightarrow (L \vee L \vee L) \\ L &\rightarrow id \mid \neg id \end{aligned}$$

AGGIUNGHIAMO UN NUOVO ASSIOMA

$$F' \rightarrow \cdot F \$$$

INIZIAMO ORA CON LA SCRITTURA DEGLI STATI



SAREMO ORA PRONTI PER SCRIVERE LE TAVOLE ACTION E GOTO:

TAVOLA ACTION

	(	V	$\wedge$	id	$\neg$	\$
0						
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11	ERR	ERR	SHIFTA	ERR	TAUL	ERR
12	ERR	ERR	ERR	ERR	SHIFTB	SHIFTA
13						
14						
15						

TAVOLA GOTO

F	F'	C	L