Come descrivere un Linguaggio di Programmazione

Sommario: 7 marzo, 2017

- Sintassi, Semantica, Pragmatica e Implementazione
- Grammatiche e Sintassi
- Grammatiche Context Free: Derivazione, Alberi, Ambiguità
- Sintassi: Proprietà contestuali
- Compilatore: Le fasi di analisi sintattica, semantica e di generazione del codice
- Semantica: Formalismi
- Semantica operazionale strutturata: Stato, Transizione, Computazione.

Sintassi

- Un programma si presenta come una "lunga" sequenza di caratteri che rappresentano simboli e strutture del linguaggio con cui il programma è espresso.

```
#include<stdio.h> #include<stdlib.h> voidXSwap(intA[], intB[]){A[0] = ...
```

- **Lessico**. La definizione di come e quali sequenze di caratteri formano simboli (inclusi i separatori di simboli) affidata al Lessico.

- **Sintassi**. La definizione di come e quali sequenze di simboli formano costrutti e la struttura del programma affidata alla Sintassi.

Grammatiche

- **Grammatiche**. Lessico e Sintassi possono essere completamente definite mediante grammatiche:
  - Lineari, per il lessico, $\mathcal{G}^{\mathcal{R}}$.
  - Libere da contesto (Context Free), per la sintassi, $\mathcal{G}^{\mathcal{F}}$.
  - $\mathcal{G}^{\mathcal{R}} \subset \mathcal{G}^{\mathcal{F}}$.

- Sia A insieme finito di simboli (caratteri):
  - $\mathbf{A}^\star$ insieme di tutte le sequenze finite di simboli su A.
  - $\mathtt{A}^\star$ è chiuso rispetto all'operatore binario, associativo, $\_.\_$, chiamato *giustapposizione*[1]: esempio Pi.sa = Pisa.
  - $\epsilon \in \mathtt{A}^\star$ seq. vuota: $\epsilon.\mathtt{q} = \mathtt{q} = \mathtt{q}.\epsilon, \forall \mathtt{q} \in \mathtt{A}^\star$

Definition (Linguaggio (formale) su A)

Un linguaggio (formale) su A è un sottoinsieme di $\mathtt{A}^\star$

[1] la sua notazione è omessa quando non vi è ambiguità

Grammatiche Libere, $\mathcal{G}^{\mathcal{F}}$ /1

Definition (Grammatica Libera)

Una grammatica libera è una quadrupla (NT,T,R,S) tale che:

- **NT** insieme finito dei non terminali.
- **T** insieme finito dei terminali.
- **S** simbolo iniziale, S∈NT.
- **R** insieme finito delle produzioni,

$$\mathtt{R} \equiv \{\mathtt{V_i} \longrightarrow \mathtt{w_i} \mid \mathtt{w_i} \in (\mathtt{T} \cup \mathtt{NT})^*, \mathtt{i} \in [1, \mathtt{k}]\}$$

Esempio

$\mathtt{G} = (\{\mathtt{E}\}, \{\mathtt{I}, +, *, (,)\}, \mathtt{E}, \mathtt{R})$

$\mathtt{R} = \{\mathtt{E} \longrightarrow \mathtt{E} + \mathtt{E}, \mathtt{E} \longrightarrow \mathtt{E} * \mathtt{E}, \mathtt{E} \longrightarrow \mathtt{I}, \mathtt{E} \longrightarrow (\mathtt{E})\}$

Notazione. $\mathtt{V} \longrightarrow \alpha_1|...|\alpha_\mathtt{k}$ invece di $\mathtt{V_1} \longrightarrow \alpha_1, ..., \mathtt{V_k} \longrightarrow \alpha_\mathtt{k}$ quando, $\mathtt{V} = \mathtt{V_1} = ... = \mathtt{V_k}$.

Esempio

$\mathtt{R} = \{\mathtt{E} \longrightarrow \mathtt{E} + \mathtt{E} \mid \mathtt{E} * \mathtt{E} \mid \mathtt{I} \mid (\mathtt{E})\}$

Grammatiche Libere, $\mathcal{G}^{\mathcal{F}}$ /2

Esempio

$\mathtt{G} = (\{\mathtt{E}\}, \{\mathtt{I}, +, *, (,)\}, \mathtt{E}, \mathtt{R})$

$\mathtt{R} = \{\mathtt{E} \to \mathtt{E} + \mathtt{E}, \mathtt{E} \to \mathtt{E} * \mathtt{E}, \mathtt{E} \to \mathtt{I}, \mathtt{E} \to (\mathtt{E})\}$

Qual'è il significato di G? [2]

Definition (La relazione $\Rightarrow$)

Ogni grammatica libera $\mathtt{G} \equiv (\mathtt{NT}, \mathtt{T}, \mathtt{R}, \mathtt{S})$ definisce una relazione binaria $\Rightarrow_{\mathtt{G}}$ (o, semplicemente $\Rightarrow$) su $(\mathtt{T} \cup \mathtt{NT})^*$ tale che:

$$\forall\, \alpha.\mathtt{V}.\rho,\ \alpha.\gamma.\rho \in (\mathtt{T} \cup \mathtt{NT})^*, \quad \alpha\mathtt{V}\rho \Rightarrow \alpha\gamma\rho \text{ sse } \mathtt{V} \to \gamma \in \mathtt{R}$$

Definition (Linguaggio generato)

Sia $\mathtt{G} \equiv (\mathtt{NT}, \mathtt{T}, \mathtt{R}, \mathtt{S})$ libera. Il linguaggio definito da $\mathtt{G}$:

$$\mathcal{L}(\mathtt{G}) = \{\alpha \in \mathtt{T}^\star | \mathtt{S} \Rightarrow^\star \alpha\}$$

dove, $\Rightarrow^\star$ è la chiusura transitiva (e riflessiva) di $\Rightarrow_{\mathtt{G}}$

[2] "sse" sta per "se e solo se"

Grammatiche Libere, $\mathcal{G}^{\mathcal{F}}$ /3

Definition (Derivazione)

Sia $\mathtt{G} \equiv (\mathtt{NT}, \mathtt{T}, \mathtt{R}, \mathtt{S})$ libera. Siano $\mathtt{v}, \mathtt{w} \in (\mathtt{T} \cup \mathtt{NT})^*$. Sia $\mathtt{v} \Rightarrow^\star \mathtt{w}$. Una derivazione, in $\mathtt{G}$, da $\mathtt{v}$ a $\mathtt{w}$, è una sequenza $\mathtt{v} \Rightarrow \mathtt{w}_0 \Rightarrow \mathtt{w}_1 \Rightarrow \ldots \Rightarrow \mathtt{w}$

Esempio

La derivazione ci fornisce una prova di appartenenza. Si dimostri che $\mathtt{I} * \mathtt{I} + \mathtt{I} \in \mathcal{L}(\mathtt{G})$.

$\mathtt{G} = (\{\mathtt{E}\}, \{\mathtt{I}, +, *\}, \mathtt{E}, \mathtt{R})$

$\mathtt{R} = \{\mathtt{E} \longrightarrow \mathtt{E} + \mathtt{E}, \mathtt{E} \longrightarrow \mathtt{E} * \mathtt{E}, \mathtt{E} \longrightarrow \mathtt{I}, \mathtt{E} \longrightarrow (\mathtt{E})\}$

Procediamo applicando una $\Rightarrow$ alla volta a partire da $\mathtt{E}$

$\mathtt{E} \Rightarrow_2 \mathtt{E} * \mathtt{E}$ [3]

$\Rightarrow_3 \mathtt{I} * \mathtt{E}$

$\Rightarrow_1 \mathtt{I} * \mathtt{E} + \mathtt{E}$

$\Rightarrow_3 \mathtt{I} * \mathtt{I} + \mathtt{E}$

$\Rightarrow_3 \mathtt{I} * \mathtt{I} + \mathtt{I}$

Aggiungendo induzione possiamo dimostrare che:

$$\{\mathtt{I}.\alpha_1.\cdots.\alpha_\mathtt{n} \mid \alpha_\mathtt{i} \in \{+\mathtt{I}, *\mathtt{I}\}, \mathtt{n} \geq 0\} \subset \mathcal{L}(G).$$

[3] Un pedice può indicare la produzione usata. In questo caso, usiamo la posizione a partire da 1 per la produzione più a sinistra

Grammatiche Libere: La relazione $\Rightarrow$ su alberi /1

Esempio

$$G = (\{E\}, \{I, +, *, (,)\}, E, R)$$
$$R = \{E \to E + E, E \to E * E, E \to I, E \to (E)\}$$

- La $\Rightarrow$ su sequenze di simboli (stringhe) grammaticali ci dice proprio tutto?

$$E \Rightarrow_2 E * E \Rightarrow_3 I * E \Rightarrow_1 I * E + E \Rightarrow_3 I * I + E \Rightarrow_3 I * I + I$$
$$E \Rightarrow_2 E * E \Rightarrow_1 E * E + E \Rightarrow_3 I * E + E \Rightarrow_3 I * I + E \Rightarrow_3 I * I + I$$
$$E \Rightarrow_1 E + E \Rightarrow_2 E * E + E \Rightarrow_3 I * E + E \Rightarrow_3 I * I + E \Rightarrow_3 I * I + I$$

- In realtà NO.
  - 2 delle 3 derivazioni sopra ci dicono la stessa cosa, 1 dice una cosa.
  - Quali? In cosa, sono diverse?
  - Useremo la struttura degli alberi per rispondere.

- La sequenza di simboli grammaticali non è abbastanza espressiva

Grammatiche Libere: La relazione $\Rightarrow$ su alberi /2

Definition

- $[] \in \mathtt{Tree}^{\mathtt{A}}$
- $[\mathtt{a} - (\mathtt{t}_1, ..., \mathtt{t}_n)] \in \mathtt{Tree}^{\mathtt{A}}$, per $\mathtt{a} \in \mathtt{A}, \mathtt{t}_i \in \mathtt{Tree}^{\mathtt{A}}, \mathtt{n} \geq 0$

Notazione. [a - ()] può essere scritto [a].

- Gli alberi possono essere utilizzati come relazioni sulla struttura di un termine
  - Ogni arco è una coppia della relazione *sottotermine*
- Hanno una rappresentazione grafica.
  - mostriamo quella di [E - ([E],[+],[E])]

Grammatiche Libere: La relazione $\Rightarrow$ su alberi /3

Definition (La relazione $\Rightarrow^{\mathtt{Tree}}$)

Ogni grammatica libera $\mathtt{G} \equiv (\mathtt{NT}, \mathtt{T}, \mathtt{R}, \mathtt{S})$ definisce una relazione binaria $\Rightarrow^{\mathtt{Tree}}_{\mathtt{G}}$ (o, semplicemente $\Rightarrow^{\mathtt{Tree}}$, ovvero $\Rightarrow$) su $\mathtt{Tree}^{\mathtt{T} \cup \mathtt{NT}}$ tale che:

- $[\mathtt{V}] \Rightarrow [\mathtt{V} - ([\mathtt{u}_1], ..., [\mathtt{u}_n])]$
  sse $\mathtt{V} \longrightarrow \mathtt{u}_1 . \ldots . \mathtt{u}_n \in \mathtt{R}$ and $\mathtt{u}_i \in (\mathtt{T} \cup \mathtt{NT})$
- $[\mathtt{V} - (\mathtt{t}_1, ..., \mathtt{t}_j, ..., \mathtt{t}_n)] \Rightarrow [\mathtt{V} - (\mathtt{t}_1, ..., \mathtt{r}_j, ..., \mathtt{t}_n)]$
  sse $1 \leq j \leq n \wedge (\forall i) \mathtt{t}_i \in \mathtt{Tree}^{\mathtt{T} \cup \mathtt{NT}} \wedge \mathtt{t}_j \Rightarrow \mathtt{r}_j$

Definition (Linguaggio generato su $\mathtt{Tree}^{\mathtt{T} \cup \mathtt{NT}}$ - Parse Tree set)

Sia $\mathtt{G} \equiv (\mathtt{NT}, \mathtt{T}, \mathtt{R}, \mathtt{S})$ libera. Il linguaggio definito da $\mathtt{G}$:

$$\mathcal{L}(\mathtt{G}) = \{\alpha \in \mathtt{Tree}^{\mathtt{T} \cup \mathtt{NT}} \mid [\mathtt{S}] \Rightarrow^{\star} \alpha\}$$

dove, $\Rightarrow^{\star}$ è la chiusura transitiva (e riflessiva) di $\Rightarrow$

Grammatiche Libere: La relazione $\Rightarrow$ su alberi /4

Definition (Derivazione su $\mathtt{Tree}^{\mathtt{T}\cup\mathtt{NT}}$)

Sia $\mathtt{G} \equiv (\mathtt{NT}, \mathtt{T}, \mathtt{R}, \mathtt{S})$ libera. Siano $\mathtt{u}, \mathtt{q} \in \mathtt{Tree}^{\mathtt{T}\cup\mathtt{NT}}$. Sia $\mathtt{u} \Rightarrow^{\star} \mathtt{q}$. Una derivazione, in $\mathtt{G}$, da $\mathtt{u}$ a $\mathtt{q}$, è una sequenza

$\mathtt{u} \Rightarrow \mathtt{q}_0 \Rightarrow \mathtt{q}_1 \Rightarrow \ldots \Rightarrow \mathtt{q}$

Esempio

$\mathtt{G} = (\{\mathtt{E}\}, \{\mathtt{I}, +, *, (,)\}, \mathtt{E}, \mathtt{R})$

$\mathtt{R} = \{\mathtt{E} \rightarrow \mathtt{E}+\mathtt{E}, \mathtt{E} \rightarrow \mathtt{E}*\mathtt{E}, \mathtt{E} \rightarrow \mathtt{I}, \mathtt{E} \rightarrow (\mathtt{E})\}$

- La $\Rightarrow$ su stringhe ci dice proprio tutto?

$\mathtt{E} \Rightarrow_2 \mathtt{E}*\mathtt{E} \Rightarrow_3 \mathtt{I}*\mathtt{E} \Rightarrow_1 \mathtt{I}*\mathtt{E}+\mathtt{E} \Rightarrow_3 \mathtt{I}*\mathtt{I}+\mathtt{E} \Rightarrow_3 \mathtt{I}*\mathtt{I}+\mathtt{I}$

$\mathtt{E} \Rightarrow_2 \mathtt{E}*\mathtt{E} \Rightarrow_1 \mathtt{E}*\mathtt{E}+\mathtt{E} \Rightarrow_3 \mathtt{I}*\mathtt{E}+\mathtt{E} \Rightarrow_3 \mathtt{I}*\mathtt{I}+\mathtt{E} \Rightarrow_3 \mathtt{I}*\mathtt{I}+\mathtt{I}$

$\mathtt{E} \Rightarrow_1 \mathtt{E}+\mathtt{E} \Rightarrow_2 \mathtt{E}*\mathtt{E}+\mathtt{E} \Rightarrow_3 \mathtt{I}*\mathtt{E}+\mathtt{E} \Rightarrow_3 \mathtt{I}*\mathtt{I}+\mathtt{E} \Rightarrow_3 \mathtt{I}*\mathtt{I}+\mathtt{I}$

- La $\Rightarrow$ su alberi?

$[\mathtt{E}] \Rightarrow_2 [\mathtt{E}-([\mathtt{E}],[*],[\mathtt{E}])] \Rightarrow_3 [\mathtt{E}-([\mathtt{E}-([\mathtt{I}])],[*],[\mathtt{E}])]$

$\Rightarrow^{\star} [\mathtt{E}-([\mathtt{E}-([\mathtt{I}])],[*],[\mathtt{E}-([\mathtt{E}-([\mathtt{I}])],[+],[\mathtt{E}-([\mathtt{I}])])])]$

$[\mathtt{E}] \Rightarrow^{\star} \ldots$

$[\mathtt{E}] \Rightarrow_1 [\mathtt{E}-([\mathtt{E}],[+],[\mathtt{E}])] \Rightarrow_2 [\mathtt{E}-([\mathtt{E}-([\mathtt{E}],[*],[\mathtt{E}]],[+],[\mathtt{E}])]$

$\Rightarrow^{\star} [\mathtt{E}-([\mathtt{E}-([\mathtt{E}-([\mathtt{I}])],[*],[\mathtt{E}-([\mathtt{I}])]],[+],[\mathtt{E}-([\mathtt{I}])])]$

La relazione $\Rightarrow$ su alberi: Una vista grafica

Esempio

- La $\Rightarrow$ su alberi?

$$[\mathtt{E}] \Rightarrow_2 [\mathtt{E} - ([\mathtt{E}], [*], [\mathtt{E}])] \Rightarrow_3 [\mathtt{E} - ([\mathtt{E} - ([\mathtt{I}])], [*], [\mathtt{E}])]$$
$$\Rightarrow^\star [\mathtt{E} - ([\mathtt{E} - ([\mathtt{I}])], [*], [\mathtt{E} - ([\mathtt{E} - ([\mathtt{I}])], [+], [\mathtt{E} - ([\mathtt{I}])])])]$$

$$[\mathtt{E}] \Rightarrow^\star \ldots$$
$$[\mathtt{E}] \Rightarrow_1 [\mathtt{E} - ([\mathtt{E}], [+], [\mathtt{E}])] \Rightarrow_2 [\mathtt{E} - ([\mathtt{E} - ([\mathtt{E}], [*], [\mathtt{E}]], [+], [\mathtt{E}])]$$
$$\Rightarrow^\star [\mathtt{E} - ([\mathtt{E} - ([\mathtt{E} - ([\mathtt{I}])], [*], [\mathtt{E} - ([\mathtt{I}])]], [+], [\mathtt{E} - ([\mathtt{I}])])]$$

Grammatiche Libere: Frontiera di un albero /5

Definition (Albero di derivazione sintattica o Parse Tree)

Sia $\mathtt{G} \equiv (\mathtt{NT}, \mathtt{T}, \mathtt{R}, \mathtt{S})$ libera. Sia $\mathtt{Q_G} = \{\mathtt{q} \in \mathtt{Tree}^{\mathtt{T} \cup \mathtt{NT}} \mid [\mathtt{S}] \Rightarrow^{\star} \mathtt{q}\}$. $\mathtt{Q_G}$ è l'insieme di tutti e soli gli alberi di derivazione sintattica, detti anche, parse tree, di $\mathtt{G}$.

Definition (Frontiera di un albero /1)

La frontiera di un albero $\mathtt{q} \in \mathtt{Tree}^{\mathtt{A}}$ è una stringa di $\mathtt{A}^{\star}$, ovvero $\mathtt{fron}(\mathtt{q}) \in \mathtt{A}^{\star}$ per ogni $\mathtt{q}$, così definita:

- $\mathtt{fron}([]) = \epsilon$
- $\mathtt{fron}([\mathtt{a}]) = \mathtt{a}$
- $\mathtt{fron}([\mathtt{a} - (\mathtt{t_1}, ..., \mathtt{t_n})]) = \mathtt{fron}(\mathtt{t_1}).\cdots.\mathtt{fron}(\mathtt{t_n}), \quad \mathtt{n} > 0$

Definition (Frontiera di un albero)

La frontiera di un albero $\mathtt{q} \in \mathtt{Tree}^{\mathtt{A}}$ è una stringa di $\mathtt{A}^\star$, ovvero $\mathtt{fron(q)} \in \mathtt{A}^\star$ per ogni q, così definita:

- $\mathtt{fron}([]) = \epsilon$
- $\mathtt{fron}([\mathtt{a}]) = \mathtt{a}$
- $\mathtt{fron}([\mathtt{a} - (\mathtt{t}_1, ..., \mathtt{t}_n)]) = \mathtt{fron}(\mathtt{t}_1).\cdots.\mathtt{fron}(\mathtt{t}_n), \quad \mathtt{n} > 0$

Esempio

$$\mathtt{fron}([\mathtt{E} - ([\mathtt{E} - ([\mathtt{I}])], [*], [\mathtt{E} - ([\mathtt{E} - ([\mathtt{I}])], [+], [\mathtt{E} - ([\mathtt{I}])])])]) =$$
$$= \mathtt{fron}([\mathtt{E} - ([\mathtt{I}])]).\mathtt{fron}([*]).\mathtt{fron}([\mathtt{E} - ([\mathtt{E} - ([\mathtt{I}])], [+], [\mathtt{E} - ([\mathtt{I}])])])$$
$$= \mathtt{fron}([\mathtt{I}]).\mathtt{fron}([*]).\mathtt{fron}([\mathtt{E} - ([\mathtt{I}])]).\mathtt{fron}([+]).\mathtt{fron}([\mathtt{E} - ([\mathtt{I}])])$$
$$= \mathtt{fron}([\mathtt{I}]).\mathtt{fron}([*]).\mathtt{fron}([\mathtt{I}]).\mathtt{fron}([+]).\mathtt{fron}([\mathtt{I}])$$
$$= \mathtt{I}.*.\mathtt{I}.+.\mathtt{I} = \mathtt{I} * \mathtt{I} + \mathtt{I}$$

Grammatiche Libere: Ambiguità /6

Definition (Linguaggio generato utilizzando Parse Tree)

Sia $\mathtt{G} \equiv (\mathtt{NT}, \mathtt{T}, \mathtt{R}, \mathtt{S})$ libera. Il linguaggio definito da $\mathtt{G}$:

$$\mathcal{L}(\mathtt{G}) = \{\mathtt{fron(q)} \in \mathtt{T}^\star \mid [\mathtt{S}] \Rightarrow^\star \mathtt{q}\}$$

dove, $\Rightarrow^\star$ è la chiusura transitiva (e riflessiva) di $\Rightarrow_{\mathtt{G}}^{\mathtt{Tree}^{\mathtt{T} \cup \mathtt{NT}}}$

Definition (Grammatica Ambigua)

$\mathtt{G} \equiv (\mathtt{NT}, \mathtt{T}, \mathtt{R}, \mathtt{S})$ è ambigua se $\exists \mathtt{q}_1 \neq \mathtt{q}_2 \in \mathtt{Tree}^{\mathtt{T} \cup \mathtt{NT}}$:

- $[\mathtt{S}] \Rightarrow^\star \mathtt{q}_1, [\mathtt{S}] \Rightarrow^\star \mathtt{q}_2$
- $\mathtt{fron}([\mathtt{q}_1]) = \mathtt{fron}([\mathtt{q}_2]) \in \mathtt{T}^*$

Grammatiche Libere: Ambiguità /7

- La **sintassi di un LP** deve essere espressa mediante una grammatica libera **non ambigua**
- Ogni **termine** di LP deve avere **un'unica struttura sintattica** che ne mostra i sottotermini componenti

Esempio Grammatica Ambigua

$$G = (\{E\}, \{I, +, *\}, E, R)$$
$$R = \{E \longrightarrow E + E, E \longrightarrow E * E, E \longrightarrow I, E \longrightarrow (E)\}$$

Esempio Grammatica non Ambigua

$$G' = (\{E, F, T\}, \{I, +, *\}, E, R')$$
$$R' = \{E \longrightarrow E + F \mid F$$
$$F \longrightarrow F * T \mid T$$
$$T \longrightarrow I \mid (E)\}$$

$$\mathcal{L}(G) = \mathcal{L}(G')$$

Grammatiche: Lessico e Sintassi

- Lessico e Sintassi si integrano nella definizione di un Linguaggio

Esempio sottoLinguaggio delle Espressioni Aritmetiche
Sintassi: Espressione

$$\mathtt{G} = (\{\mathtt{E}, \mathtt{F}, \mathtt{T}\}, \{\mathtt{N}, +, -, *\}, \mathtt{E}, \mathtt{R_G})$$

$$\begin{aligned}\mathtt{R_G} = \{\ & \mathtt{E} \longrightarrow \mathtt{E} + \mathtt{F} \mid \mathtt{F} \\ & \mathtt{F} \longrightarrow \mathtt{F} * \mathtt{T} \mid \mathtt{T} \\ & \mathtt{T} \longrightarrow -\mathtt{T} \mid \mathtt{N} \mid (\mathtt{E})\}\end{aligned}$$

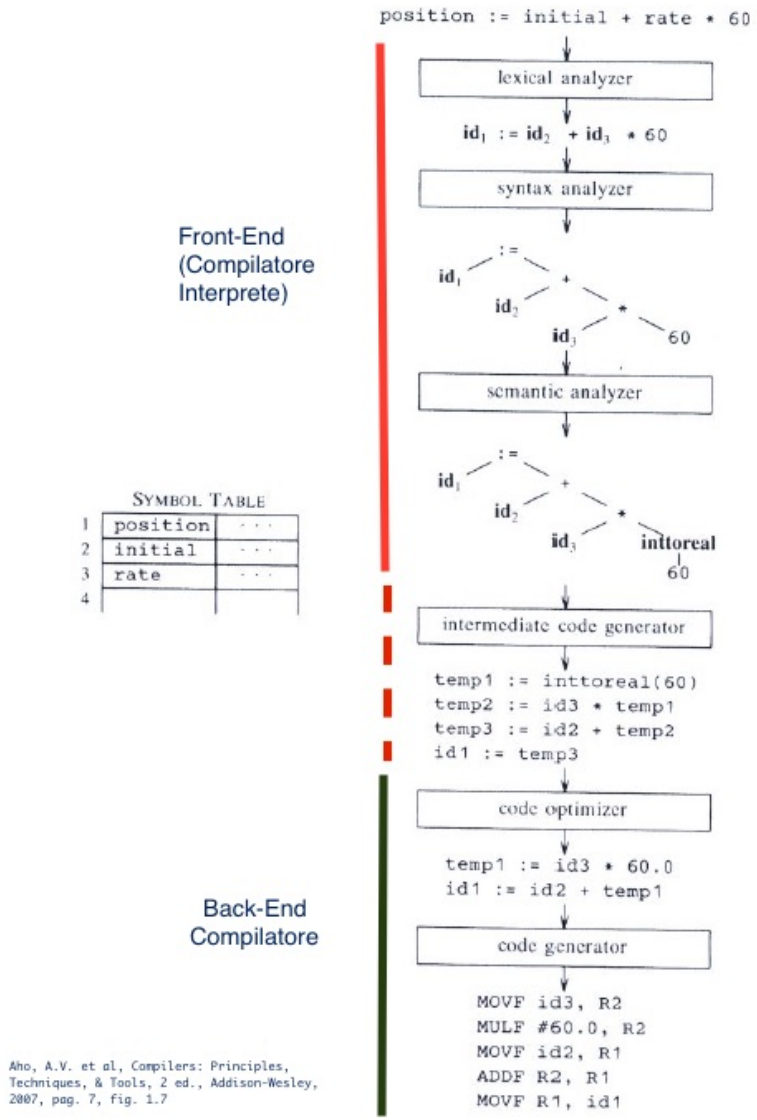
Lessico: Naturale (Identificatore)

$$\mathtt{H} = (\{\mathtt{L}, \mathtt{N}, \mathtt{D}\}, \{-, +, *, 0, \cdots, 9\}, \mathtt{L}, \mathtt{R_H})$$

$$\begin{aligned}\mathtt{R_H} = \{\ & \mathtt{L} \longrightarrow \mathtt{N} \mid * \mid - \mid + \\ & \mathtt{N} \longrightarrow \mathtt{D}\ \mathtt{N} \mid \mathtt{D} \\ & \mathtt{D} \longrightarrow 0 \mid 1 \mid \cdots \mid 9\}\end{aligned}$$

Riconosciamo la stringa, 13*-5+007$\in \mathcal{L}(<\mathtt{G},\mathtt{H}>)$.

Compilatore/Interprete: Front-End e Back-End



Aho, A.V. et al, Compilers: Principles, Techniques, & Tools, 2 ed., Addison-Wesley, 2007, pag. 7, fig. 1.7

Vincoli Sintattici contestuali: Problema

- **Programmi (sintatticamente) Legali** sono i programmi sintatticamente corretti a cui la semantica del linguaggio può associare la funzione calcolabile descritta dal programma.

- La sintassi espressa da una grammatica non ambigua non è sufficiente a identificare tutti e soli i programmi legali.

- Ad esempio: `x = 7` è un assegnamento
  - sintatticamente corretto in tutti i programmi C [4]
  - ma non è legale se `x` non è stato *dichiarato* [5]

[4] e di tutti i L.P. dove `x`, `_=_`, `7` siano la sintassi per una variabile, per l'operatore di assegnamento, per un'espressione

[5] opportunamente, nel programma

Vincoli Sintattici contestuali: Soluzione

- **Proprietà contestuali** (delle regole di composizione) non possono essere espresse da una grammatica libera (da contesto, i.e. Context-Free):
  - identificatori usati devono essere dichiarati prima ...;
  - numero di parametri attuali e formali devono coincidere
  - compatibilità nell'assegnamento, tra il tipo di una variabile e il tipo dell'espressione assegnata
  - ...
- Queste proprietà contestuali **fanno parte della definizione del linguaggio**, ma devono essere espressi con strumenti adatti.
- **Analisi Statica** Sono procedimenti di analisi e formalismi specifici (ad es., sistema dei tipi) per controllare che il programma soddisfi tutti le propriet contestuali del linguaggio.

Compilatore, Interprete: Front-End

- **Programmi (sintatticamente) Legali.** Compilatori e Interpreti trattano tutti questi aspetti nel front-end.
- Il **Front-End** di C./I.[6] provvede alla:
  - **Analisi Lessicale** in accordo al lessico
  - **Analisi Sintattica** in accordo alla sintassi
  - **Analisi Statica** in accodo alle proprietà contestuali
  - **Costruzione Abstract Tree** `AT(p)` che fornisce una rappresentazione interna del programma sorgente `p`.

- `AT(p)` è successivamente utilizzato dal **Back-End** di C./I. per completare la realizzazione dell'esecutore.
- Vediamo tutto questo nella tipica struttura (in fasi) di un C.

[6] Compilatori e Interpreti

Semantica: Denotazionale e Operazionale

- Associare significato ai termini del linguaggio.
- Vari formalismi con differenti accezioni di significato e differenti usi.
- Due da ricordare:
  - **Semantica Denotazionale**
    - *Significato*: Funzione Calcolata
    - *Usi*: Molteplici incluso definizione di C. e di I.
    - *Laboratorio*: Interpreti di $\mathcal{L}$, derivabili dalla S. Den. di $\mathcal{L}$, riscritta in un Linguaggio di P. Funzionale, OCaml
    - *Caratteristica*: Orientata allo studio di proprietà di $\mathcal{L}$, Astratta dall'implementazione di $\mathcal{L}$.
  - **Semantica Operazionale**

Semantica: Operazionale

- Associare significato ai termini del linguaggio.
- Vari formalismi con differenti accezioni di significato e differenti usi.
- Due da ricordare:
  - **Semantica Denotazionale**
  - **Semantica Operazionale**
    - *Significato*: Computazione (su Macchina)
    - *Usi*: Definizione di I., anche didattici e per studio di programmi e programmazione in $\mathcal{L}$
    - *Caratteristica*: Basata sulla nozione di Stato, transizione di stato, computazione
- Useremo una Semantica Operazionale *Strutturata* in cui la Macchina è una Macchina Astratta.

Semantica Operazionale Strutturata, SOS

- La definizione di Stato, Transizione di stato e Computazione di una S.O.S, dipende dal Linguaggio.
- Vediamole per il Linguaggio sotto.
- Assumiamo che sintassi (concreta) e lessico siano già stati trattati, analizzati, fornendoci la sintassi astratta dei termini del linguaggio: "(AExp-Aexp)" va letto come un albero con radice "-" e due alberi "AExp" come figli (i simboli "(",",")" rimarcano questa lettura).

$$
\begin{aligned}
Num &::= 1 \mid 2 \mid 3 \mid \ldots \\
Var &::= X_1 \mid X_2 \mid X_3 \mid \ldots
\end{aligned}
$$

$$
\begin{aligned}
AExp &::= Num \mid Var \mid (AExp + AExp) \mid (AExp - AExp) \\
BExp &::= \mathbf{tt} \mid \mathbf{ff} \mid (AExp == AExp) \mid \neg BExp \mid (BExp \wedge BExp)
\end{aligned}
$$

$$
\begin{aligned}
Com ::= {} & \mathbf{skip} \mid Var := AExp \mid Com; Com \mid \\
& \mathbf{if}\ BExp\ \mathbf{then}\ Com\ \mathbf{else}\ Com \mid \ \mathbf{while}\ BExp\ \mathbf{do}\ Com
\end{aligned}
$$

SOS: Lo stato

$$
\begin{aligned}
Num &::= 1 \mid 2 \mid 3 \mid \ldots \\
Var &::= X_1 \mid X_2 \mid X_3 \mid \ldots
\end{aligned}
$$

$$
\begin{aligned}
AExp &::= Num \mid Var \mid (AExp + AExp) \mid (AExp - AExp) \\
BExp &::= \mathbf{tt} \mid \mathbf{ff} \mid (AExp == AExp) \mid \neg BExp \mid (BExp \wedge BExp)
\end{aligned}
$$

$$
\begin{aligned}
Com &::= \mathbf{skip} \mid Var := AExp \mid Com; Com \mid \\
&\quad\ \mathbf{if}\ BExp\ \mathbf{then}\ Com\ \mathbf{else}\ Com \mid\ \mathbf{while}\ BExp\ \mathbf{do}\ Com
\end{aligned}
$$

- La definizione di Stato.
  - Memoria simbolica per trattare le variabili (valori modificabili)
  - Non abbiamo I/O, file systems,.... Lo stato è la sola memoria
  - Lo stato è rappresentato da sequenza finita di coppie $(\mathtt{X_i}, \mathtt{n_i})$, indicante ...
  - Lo stato è denotato con le variabili σ, τ (anche con pedici)

SOS: Transizione

$$
\begin{aligned}
\mathit{Num} &::= 1 \mid 2 \mid 3 \mid \ldots \\
\mathit{Var} &::= X_1 \mid X_2 \mid X_3 \mid \ldots
\end{aligned}
$$

$$
\begin{aligned}
\mathit{AExp} &::= \mathit{Num} \mid \mathit{Var} \mid (\mathit{AExp} + \mathit{AExp}) \mid (\mathit{AExp} - \mathit{AExp}) \\
\mathit{BExp} &::= \mathbf{tt} \mid \mathbf{ff} \mid (\mathit{AExp} == \mathit{AExp}) \mid \neg \mathit{BExp} \mid (\mathit{BExp} \wedge \mathit{BExp})
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Com} ::= {} & \mathbf{skip} \mid \mathit{Var} := \mathit{AExp} \mid \mathit{Com}; \mathit{Com} \mid \\
& \mathbf{if}\ \mathit{BExp}\ \mathbf{then}\ \mathit{Com}\ \mathbf{else}\ \mathit{Com} \mid\ \mathbf{while}\ \mathit{BExp}\ \mathbf{do}\ \mathit{Com}
\end{aligned}
$$

- La definizione di Transizione.
  - Esecuzione di un costrutto c nello stato σ della Macchina Astratta
  - La esprimiamo con:
    - $\langle c, \sigma \rangle \rightarrow \tau$, indicante ...
    - $\langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle$, indicante ...
    - $\langle c_1, \sigma_1 \rangle \rightarrow \langle c'_1, \sigma'_1 \rangle, \cdots, \langle c_k, \sigma_k \rangle \rightarrow \langle c'_k, \sigma'_k \rangle \ / \ \langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle$, k-premesse/1-conclusione, indicante ...
  - La semantica di $\mathcal{L}$ fornisce le transizione che sono specifiche per $\mathcal{L}$

SOS: Computazione

$$
\begin{aligned}
\mathit{Num} &::= 1 \mid 2 \mid 3 \mid \ldots \\
\mathit{Var} &::= X_1 \mid X_2 \mid X_3 \mid \ldots
\end{aligned}
$$

$$
\begin{aligned}
\mathit{AExp} &::= \mathit{Num} \mid \mathit{Var} \mid (\mathit{AExp} + \mathit{AExp}) \mid (\mathit{AExp} - \mathit{AExp}) \\
\mathit{BExp} &::= \mathbf{tt} \mid \mathbf{ff} \mid (\mathit{AExp} == \mathit{AExp}) \mid \neg \mathit{BExp} \mid (\mathit{BExp} \wedge \mathit{BExp})
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Com} &::= \mathbf{skip} \mid \mathit{Var} := \mathit{AExp} \mid \mathit{Com}; \mathit{Com} \mid \\
&\qquad \mathbf{if}\ \mathit{BExp}\ \mathbf{then}\ \mathit{Com}\ \mathbf{else}\ \mathit{Com} \mid\ \mathbf{while}\ \mathit{BExp}\ \mathbf{do}\ \mathit{Com}
\end{aligned}
$$

- La definizione di Computazione di $\mathtt{p} \in \mathcal{L}$.
  - Sequenza degli stati attraversati effettivamente dalle transizioni usate nell'esecuzione del programma p.

SOS: Notazione usata

$$
\begin{array}{lll}
\mathit{Num} & ::= & 1 \mid 2 \mid 3 \mid \ldots \\
\mathit{Var} & ::= & X_1 \mid X_2 \mid X_3 \mid \ldots
\end{array}
$$

$$
\begin{array}{lll}
\mathit{AExp} & ::= & \mathit{Num} \mid \mathit{Var} \mid (\mathit{AExp} + \mathit{AExp}) \mid (\mathit{AExp} - \mathit{AExp}) \\
\mathit{BExp} & ::= & \mathbf{tt} \mid \mathbf{ff} \mid (\mathit{AExp} == \mathit{AExp}) \mid \neg \mathit{BExp} \mid (\mathit{BExp} \wedge \mathit{BExp})
\end{array}
$$

$$
\begin{array}{lll}
\mathit{Com} & ::= & \mathbf{skip} \mid \mathit{Var} := \mathit{AExp} \mid \mathit{Com}; \mathit{Com} \mid \\
& & \mathbf{if}\ \mathit{BExp}\ \mathbf{then}\ \mathit{Com}\ \mathbf{else}\ \mathit{Com} \mid\ \mathbf{while}\ \mathit{BExp}\ \mathbf{do}\ \mathit{Com}
\end{array}
$$

- Notazione.
  $(\mathtt{X}_1, \mathtt{n}_1), \cdots, (\mathtt{X}_k, \mathtt{n}_k)$ stato con k variabili legate
  σ, τ (anche con pedici) sono stati della macchina
  $\sigma(\mathtt{X}_i) = \mathtt{n}_i$ quando $\sigma = (\mathtt{X}_1, \mathtt{n}_1), \cdots, (\mathtt{X}_i, \mathtt{n}_i), \cdots, (\mathtt{X}_k, \mathtt{n}_k)$,
  $\sigma[\mathtt{X}_i \leftarrow \mathtt{m}_i] = (\mathtt{X}_1, \mathtt{n}_1), \cdots, (\mathtt{X}_i, \mathtt{m}_i), \cdots, (\mathtt{X}_k, \mathtt{n}_k)$
  quando $\sigma = (\mathtt{X}_1, \mathtt{n}_1), \cdots, (\mathtt{X}_i, \mathtt{n}_i), \cdots, (\mathtt{X}_k, \mathtt{n}_k)$
  $\mathtt{n}, \mathtt{n}_i \in \mathtt{Num}$ valori numerici
  $\mathtt{a}, \mathtt{a}_i \in \mathtt{AExp}$ espressioni aritmetiche
  $\mathtt{b}, \mathtt{b}_i \in \mathtt{BExp}$ espressioni booleane
  tt, **ff** i valori true e false
  $\mathtt{c}, \mathtt{c}_i \in \mathtt{Com}$ comandi del linguaggio

SOS: Semantica Espressioni Aritmetiche

$$
\begin{aligned}
Num &::= 1 \mid 2 \mid 3 \mid \ldots \\
Var &::= X_1 \mid X_2 \mid X_3 \mid \ldots
\end{aligned}
$$

$$AExp ::= Num \mid Var \mid (AExp + AExp) \mid (AExp - AExp)$$

Semantica delle Espressioni `AExp`.

$$\langle X, \sigma \rangle \to \langle \sigma(X), \sigma \rangle$$

$$\frac{}{\langle (n+m), \sigma \rangle \to \langle p, \sigma \rangle} \text{ where } p = n + m \qquad \frac{}{\langle (n-m), \sigma \rangle \to \langle p, \sigma \rangle} \text{ where } p = n - m \text{ e } n \geq m$$

$$\frac{\langle a_1, \sigma \rangle \to \langle a', \sigma \rangle}{\langle (a_1 + a_2), \sigma \rangle \to \langle (a' + a_2), \sigma \rangle} \qquad \frac{\langle a_2, \sigma \rangle \to \langle a'', \sigma \rangle}{\langle (a_1 + a_2), \sigma \rangle \to \langle (a_1 + a''), \sigma \rangle}$$

$$\frac{\langle a_1, \sigma \rangle \to \langle a', \sigma \rangle}{\langle (a_1 - a_2), \sigma \rangle \to \langle (a' - a_2), \sigma \rangle} \qquad \frac{\langle a_2, \sigma \rangle \to \langle a'', \sigma \rangle}{\langle (a_1 - a_2), \sigma \rangle \to \langle (a_1 - a''), \sigma \rangle}$$

- L'ordine di valutazione degli argomenti delle operazioni è inessenziale
- Se lo volessimo da sinistra a destra: Come modificarla?

SOS: Semantica dei Comandi

$$Com ::= \textbf{skip} \mid Var := AExp \mid Com; Com \mid \textbf{if } BExp \textbf{ then } Com \textbf{ else } Com \mid \textbf{while } BExp \textbf{ do } Com$$

Semantica dei Comandi `Com`.

$$\langle \textbf{skip}, \sigma \rangle \to \sigma \quad (c1)$$

$$\langle X := n, \sigma \rangle \to \sigma[X \leftarrow n] \quad (c2) \qquad \frac{\langle a, \sigma \rangle \to \langle a', \sigma \rangle}{\langle X := a, \sigma \rangle \to \langle X := a', \sigma \rangle} \quad (c3)$$

$$\frac{\langle c_1, \sigma \rangle \to \sigma'}{\langle c_1; c_2, \sigma \rangle \to \langle c_2, \sigma' \rangle} \quad (c4) \qquad \frac{\langle c_1, \sigma \rangle \to \langle c_1', \sigma' \rangle}{\langle c_1; c_2, \sigma \rangle \to \langle c_1'; c_2, \sigma' \rangle} \quad (c5)$$

$$\langle \textbf{if tt then } c_1 \textbf{ else } c_2, \sigma \rangle \to \langle c_1, \sigma \rangle \quad (c6)$$
$$\langle \textbf{if ff then } c_1 \textbf{ else } c_2, \sigma \rangle \to \langle c_2, \sigma \rangle \quad (c7)$$

$$\frac{\langle b, \sigma \rangle \to \langle b', \sigma \rangle}{\langle \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2, \sigma \rangle \to \langle \textbf{if } b' \textbf{ then } c_1 \textbf{ else } c_2, \sigma \rangle} \quad (c8)$$

$$\langle \textbf{while } b \textbf{ do } c, \sigma \rangle \to \langle \textbf{if } b \textbf{ then } c; \textbf{while } b \textbf{ do } c \textbf{ else skip}, \sigma \rangle \quad (c9)$$

SOS: Semantica dei Comandi

$$\langle \textbf{skip}, \sigma \rangle \to \sigma \quad (c1)$$

$$\langle X := n, \sigma \rangle \to \sigma[X \leftarrow n] \quad (c2) \qquad \frac{\langle a, \sigma \rangle \to \langle a', \sigma \rangle}{\langle X := a, \sigma \rangle \to \langle X := a', \sigma \rangle} \quad (c3)$$

$$\frac{\langle c_1, \sigma \rangle \to \sigma'}{\langle c_1; c_2, \sigma \rangle \to \langle c_2, \sigma' \rangle} \quad (c4) \qquad \frac{\langle c_1, \sigma \rangle \to \langle c_1', \sigma' \rangle}{\langle c_1; c_2, \sigma \rangle \to \langle c_1'; c_2, \sigma' \rangle} \quad (c5)$$

$$\langle \textbf{if tt then } c_1 \textbf{ else } c_2, \sigma \rangle \to \langle c_1, \sigma \rangle \quad (c6)$$
$$\langle \textbf{if ff then } c_1 \textbf{ else } c_2, \sigma \rangle \to \langle c_2, \sigma \rangle \quad (c7)$$

$$\frac{\langle b, \sigma \rangle \to \langle b', \sigma \rangle}{\langle \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2, \sigma \rangle \to \langle \textbf{if } b' \textbf{ then } c_1 \textbf{ else } c_2, \sigma \rangle} \quad (c8)$$

$$\langle \textbf{while } b \textbf{ do } c, \sigma \rangle \to \langle \textbf{if } b \textbf{ then } c; \textbf{while } b \textbf{ do } c \textbf{ else skip}, \sigma \rangle \quad (c9)$$

Sia `c` la sequenza di comandi: `X := 1;` **while** $\neg$`(X == 0)` **do** `X := X - 1`

$$\begin{aligned}
&\langle c, \sigma \rangle \\
&\to \langle c', \sigma[X \leftarrow 1] \rangle \\
&\to \langle \textbf{if } \neg(X == 0) \textbf{ then } X := (X - 1); c' \textbf{ else skip}, \sigma[X \leftarrow 1] \rangle \\
&\to \langle \textbf{if } \neg(1 == 0) \textbf{ then } X := (X - 1); c' \textbf{ else skip}, \sigma[X \leftarrow 1] \rangle \\
&\to \langle \textbf{if } \neg\textbf{ff then } X := (X - 1); c' \textbf{ else skip}, \sigma[X \leftarrow 1] \rangle \\
&\to \langle \textbf{if tt then } X := (X - 1); c' \textbf{ else skip}, \sigma[X \leftarrow 1] \rangle \\
&\to \langle X := (X - 1); c', \sigma[X \leftarrow 1] \rangle \\
&\to \langle X := (1 - 1); c', \sigma[X \leftarrow 1] \rangle \\
&\to \langle X := 0; c', \sigma[X \leftarrow 1] \rangle \\
&\to \langle c', \sigma[X \leftarrow 0] \rangle \\
&\to \langle \textbf{if } \neg(X == 0) \textbf{ then } X := (X - 1); c' \textbf{ else skip}, \sigma[X \leftarrow 0] \rangle \\
&\to \langle \textbf{if } \neg(0 == 0) \textbf{ then } X := (X - 1); c' \textbf{ else skip}, \sigma[X \leftarrow 0] \rangle \\
&\to \langle \textbf{if } \neg\textbf{tt then } X := (X - 1); c' \textbf{ else skip}, \sigma[X \leftarrow 0] \rangle \\
&\to \langle \textbf{if ff then } X := (X - 1); c' \textbf{ else skip}, \sigma[X \leftarrow 0] \rangle \\
&\to \langle \textbf{skip}, \sigma[X \leftarrow 0] \rangle \\
&\to \sigma[X \leftarrow 0]
\end{aligned}$$

Esercizi. 2.10-1

- Esercizio 2.10.1
  (a) Si completi la definizione di una grammatica le cui produzioni sono sotto.
  (b) Si mostri che la grammatica ottenuta è ambigua.

$$\mathtt{E} \longrightarrow \mathtt{T} \mid \mathtt{T} + \mathtt{E} \mid \mathtt{T} - \mathtt{E}$$
$$\mathtt{T} \longrightarrow \mathtt{A} \mid \mathtt{A} * \mathtt{E}$$

Soluzione
(a) $\mathtt{G} = \ldots$
(b) Consideriamo la stringa: `s=A*A+A`. Per essa possiamo mostrare i due diversi parse tree sotto aventi come frontiera `s`

Esercizi. 2.10-4

- Esercizio 2.10.4
  Si dia una grammatica non ambigua che generi tutte e sole le sequenze di parentesi angolate bilanciate

  Soluzione

$$\mathtt{S} \longrightarrow \mathtt{B}\ \mathtt{S} \mid \epsilon$$
$$\mathtt{B} \longrightarrow \langle\ \mathtt{S}\ \rangle$$

Esercizi. 2.10-5

- Esercizio 2.10.5
  Si chiama lineare una grammatica le cui produzioni hanno la forma $\mathtt{A} \longrightarrow \mathtt{t}\ \mathtt{B}$ oppure $\mathtt{A} \longrightarrow \mathtt{t}$, dove $\mathtt{A}$, $\mathtt{B}$ siano non terminali e $\mathtt{t}$ sia un terminale o ϵ. Un linguaggio che può essere espresso con una grammatica lineare si chiama *regolare* e può essere riconosciuto da un automa a stati finiti. Si mostri che il linguaggio $\mathtt{L} = \{\mathtt{a}^{\mathtt{n}}\mathtt{b}^{\mathtt{m}} \mid \mathtt{n}, \mathtt{m} \geq 1\}$ è regolare.

  Soluzione

  $$\mathtt{S} \longrightarrow \mathtt{a}\ \mathtt{S} \mid \mathtt{B}$$
  $$\mathtt{B} \longrightarrow \mathtt{b}\ \mathtt{B} \mid \mathtt{b}$$

Esercizi. 2.10-5bis

- Esercizio 2.10.5bis
  Si dia una grammatica non ambigua per che il linguaggio $\mathtt{L} = \{\mathtt{a}^{\mathtt{n}}\mathtt{b}^{\mathtt{m}} \mid \mathtt{n} \leq \mathtt{m}\}$.

  Soluzione

  $$\mathtt{S} \longrightarrow \mathtt{a}\ \mathtt{S}\ \mathtt{b} \mid \mathtt{B}$$
  $$\mathtt{B} \longrightarrow \mathtt{b}\ \mathtt{B} \mid \epsilon$$

  od anche:

  $$\mathtt{S} \longrightarrow \mathtt{A}\ \mathtt{B}$$
  $$\mathtt{A} \longrightarrow \mathtt{a}\ \mathtt{A}\ \mathtt{b} \mid \epsilon$$
  $$\mathtt{B} \longrightarrow \mathtt{b}\ \mathtt{B} \mid \epsilon$$

Esercizi. 2.10-6bis

$$\langle X, \sigma \rangle \to \langle \sigma(X), \sigma \rangle$$

$$\langle (n+m), \sigma \rangle \to \langle p, \sigma \rangle \quad \text{where } p = n+m \qquad \langle (n-m), \sigma \rangle \to \langle p, \sigma \rangle \quad \text{where } p = n-m \text{ e } n \geq m$$

$$\frac{\langle a_1, \sigma \rangle \to \langle a', \sigma \rangle}{\langle (a_1 + a_2), \sigma \rangle \to \langle (a' + a_2), \sigma \rangle} \qquad \frac{\langle a_2, \sigma \rangle \to \langle a'', \sigma \rangle}{\langle (a_1 + a_2), \sigma \rangle \to \langle (a_1 + a''), \sigma \rangle}$$

$$\frac{\langle a_1, \sigma \rangle \to \langle a', \sigma \rangle}{\langle (a_1 - a_2), \sigma \rangle \to \langle (a' - a_2), \sigma \rangle} \qquad \frac{\langle a_2, \sigma \rangle \to \langle a'', \sigma \rangle}{\langle (a_1 - a_2), \sigma \rangle \to \langle (a_1 - a''), \sigma \rangle}$$

- Esercizio 2.10.6bis
  Si modifichino le regole delle espressioni aritmetiche in modo da prescrivere una valutazione degli argomenti da sinistra a destra per la somma e una da destra a sinistra per la sottrazione.
  Soluzione

Esercizi. 2.10-7bis

$$\langle \textbf{skip}, \sigma \rangle \to \sigma \quad (c1)$$

$$\langle X := n, \sigma \rangle \to \sigma[X \leftarrow n] \quad (c2) \qquad \frac{\langle a, \sigma \rangle \to \langle a', \sigma \rangle}{\langle X := a, \sigma \rangle \to \langle X := a', \sigma \rangle} \quad (c3)$$

$$\frac{\langle c_1, \sigma \rangle \to \sigma'}{\langle c_1; c_2, \sigma \rangle \to \langle c_2, \sigma' \rangle} \quad (c4) \qquad \frac{\langle c_1, \sigma \rangle \to \langle c_1', \sigma' \rangle}{\langle c_1; c_2, \sigma \rangle \to \langle c_1'; c_2, \sigma' \rangle} \quad (c5)$$

$$\langle \textbf{if tt then } c_1 \textbf{ else } c_2, \sigma \rangle \to \langle c_1, \sigma \rangle \quad (c6)$$
$$\langle \textbf{if ff then } c_1 \textbf{ else } c_2, \sigma \rangle \to \langle c_2, \sigma \rangle \quad (c7)$$

$$\frac{\langle b, \sigma \rangle \to \langle b', \sigma \rangle}{\langle \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2, \sigma \rangle \to \langle \textbf{if } b' \textbf{ then } c_1 \textbf{ else } c_2, \sigma \rangle} \quad (c8)$$

$$\langle \textbf{while } b \textbf{ do } c, \sigma \rangle \to \langle \textbf{if } b \textbf{ then } c; \textbf{while } b \textbf{ do } c \textbf{ else skip}, \sigma \rangle \quad (c9)$$

- Esercizio 2.10.7bis
  Siano `c` e `d` i seguenti comandi:
  `c:` `X:=1; d`
  `d:` **while**`(X==1)` **do skip**
  Si dia la sequenza di transizioni ottenute a partire dallo stato $\sigma = (\texttt{X}, 0)$
  Soluzione