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$\in \mathcal{L}(<\mathtt{G},\mathtt{H}>)$.

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

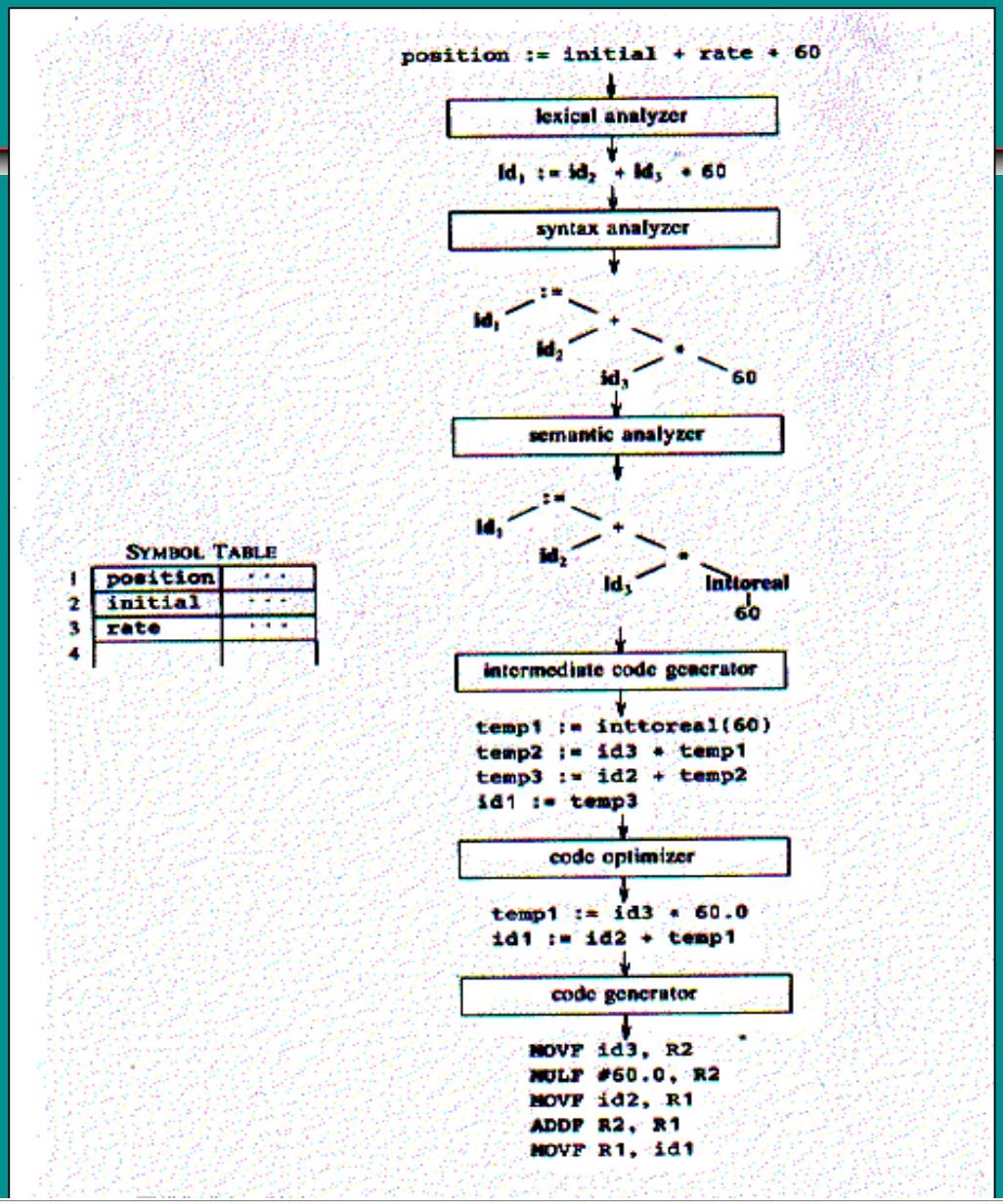
- Proprietà contestuali (delle regole di composizione) non possono essere espresse da una grammatica libera (da contesto, i.e. Context-Free):
  - identificatori usati devono essere dichiarati prima ...;
  - numero di parametri attuali e formali devono corrispondere
  - compatibilità nell'assegnamento, tra il tipo di una variabile e il tipo dell'espressione assegnata
  - ...
- Questi vincoli 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 i vincoli.

Compilatore, Interprete: Front-End

- **Programmi (sintatticamente) Legali** C. e I. trattano tutti questi aspetti nel front-end.
- Il front-end provvede alla:
  - analisi lessicale in accordo al lessico
  - analisi sintattica in accordo alla sintassi
  - analisi statica in accodo alle regole sui vincoli contestuali
  - costruzione di una rappresentazione interna del programma sorgente `p`: Abstract Tree, `AT(p)`

  del linguaggio sorgente $\mathcal{L}$
- `AT(p)` è successivamente utilizzato dal back-end di C. e/o di I. per completare la realizzazione dell'esecutore
- Vediamo tutto questo nella tipica struttura (in fasi) di un C.

from:
A.V. Aho, R. Sethi and J.D. Ullman, Compilers: Principles, Techniques and Tools, Addison-Wesley, 1988



Phases may procede in pipelining

leading to:
+ One Pass
+ Multi Pass
Compiler

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).

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

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

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

SOS: Lo stato

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

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

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

- 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

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

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

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

- 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 $\mathtt{p}$.

SOS: Notazione usata

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

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

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

- 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`

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