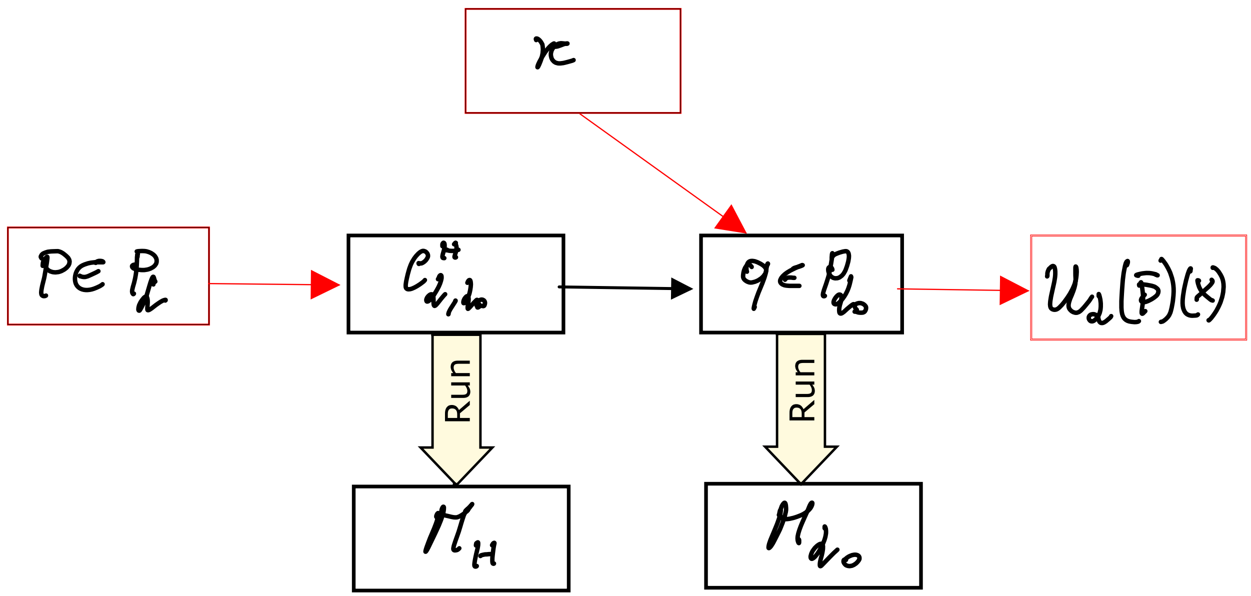
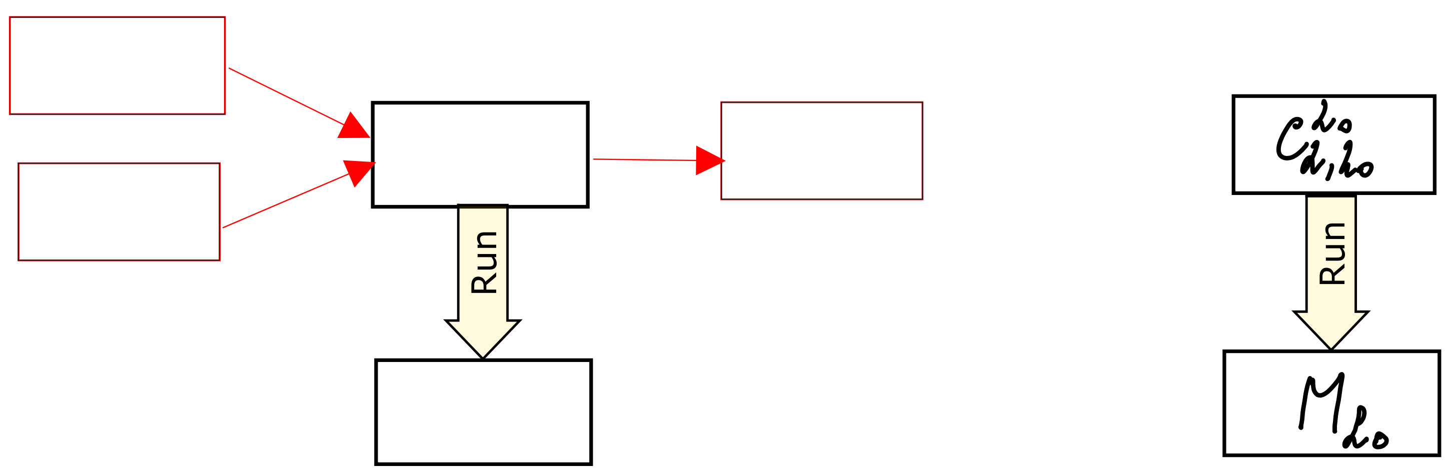
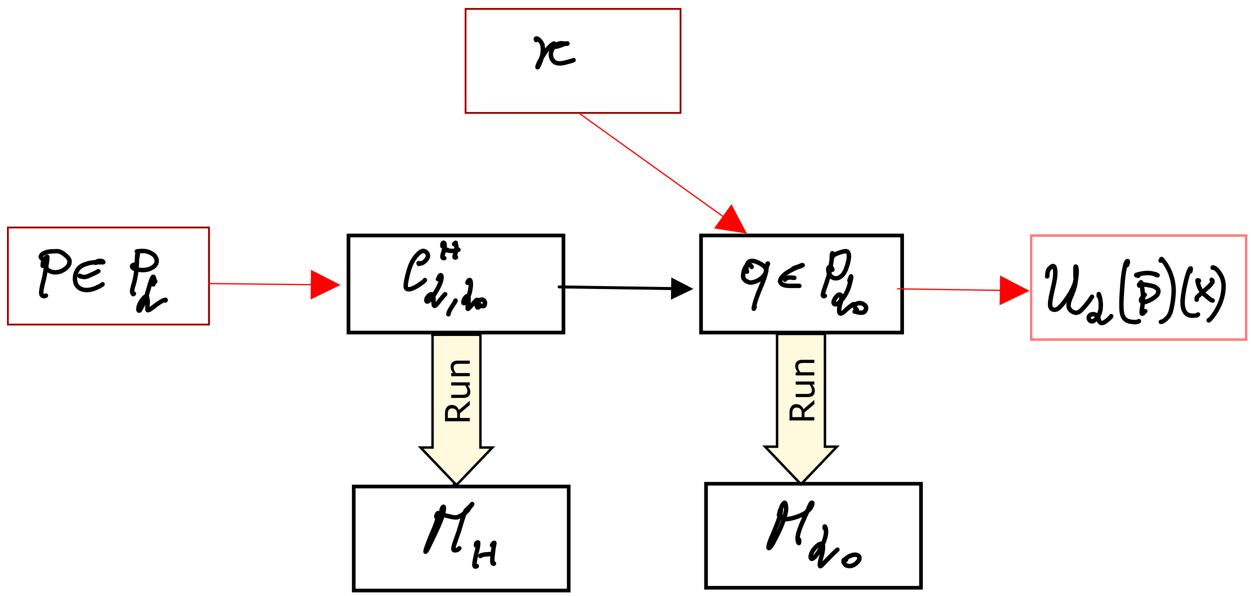
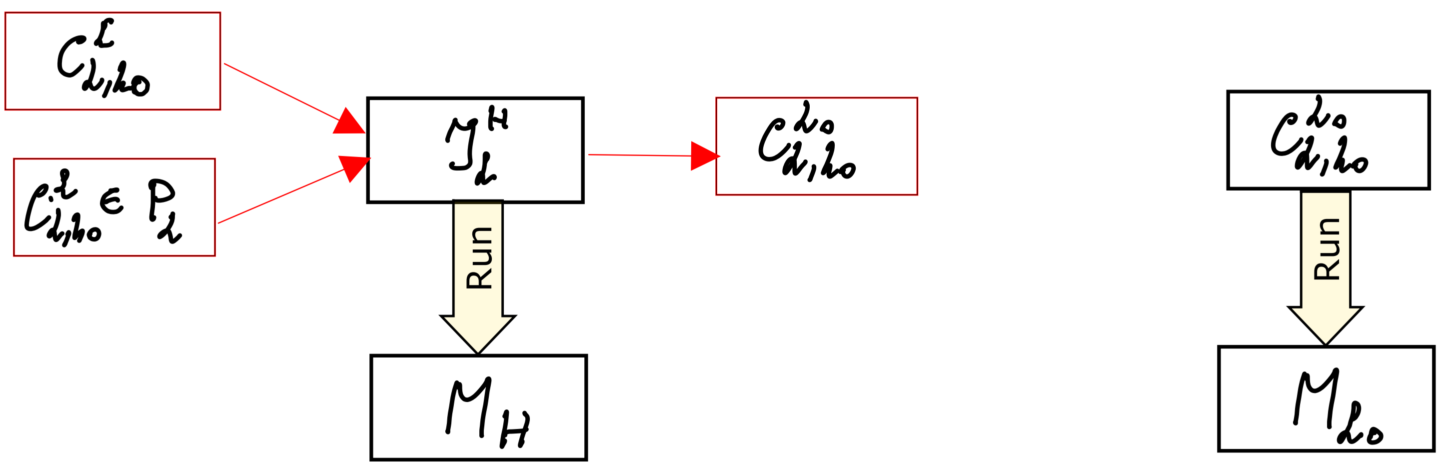
1. Costruzione di un Compilatore $C^{L_0}_{L,L_0}$ quando $L > H \gg L_0$



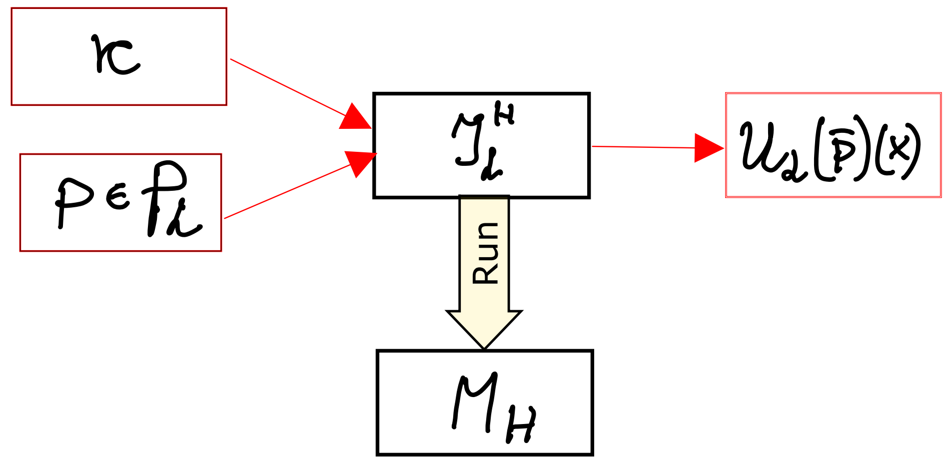


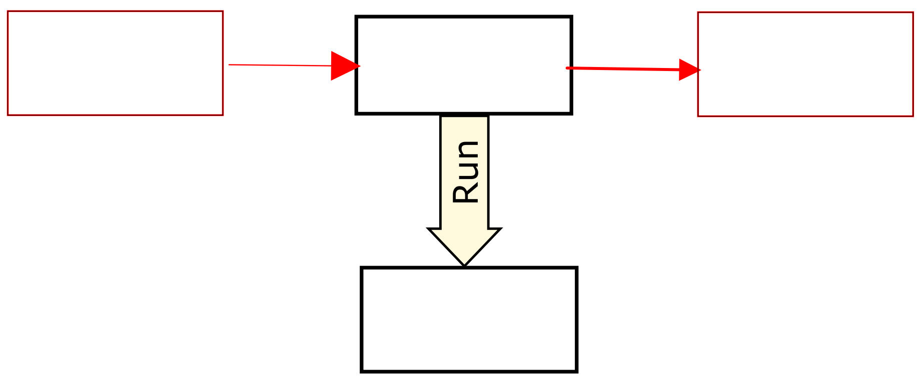
§1. Costruzione di un Compilatore $C^{L_0}_{L,L_0}$ quando $L > H \gg L_0$

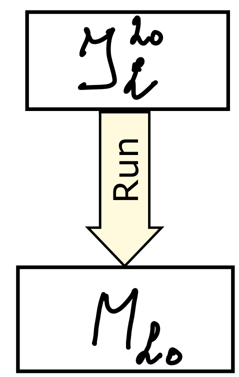




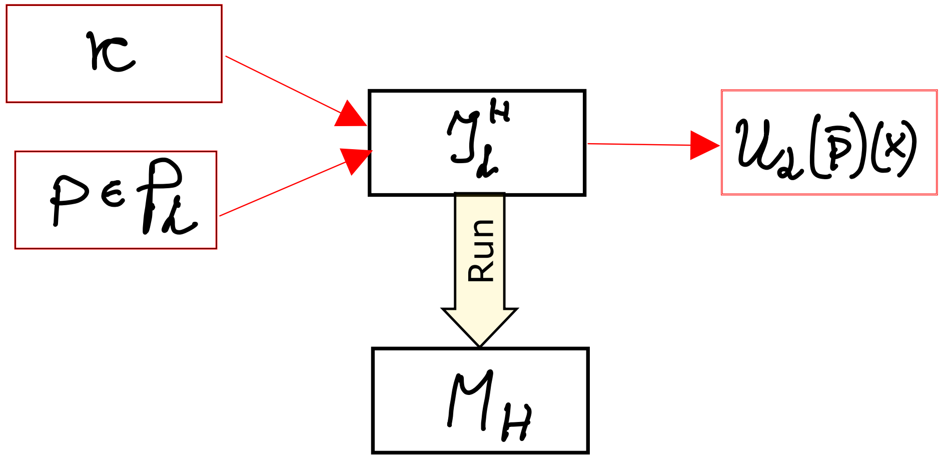
2. Costruzione di un Interprete $I_L^{L_0}$ quando $L > H \gg L_0$

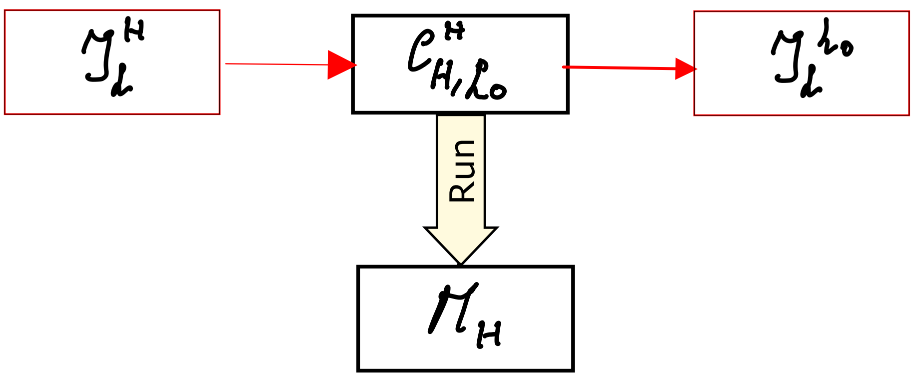


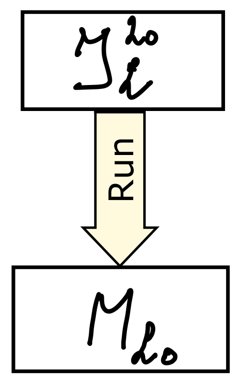




§2. Costruzione di un Interprete $\mathcal{I}_{\mathcal{L}}^{\mathcal{L}_0}$ quando $\mathcal{L} > H \gg \mathcal{L}_0$





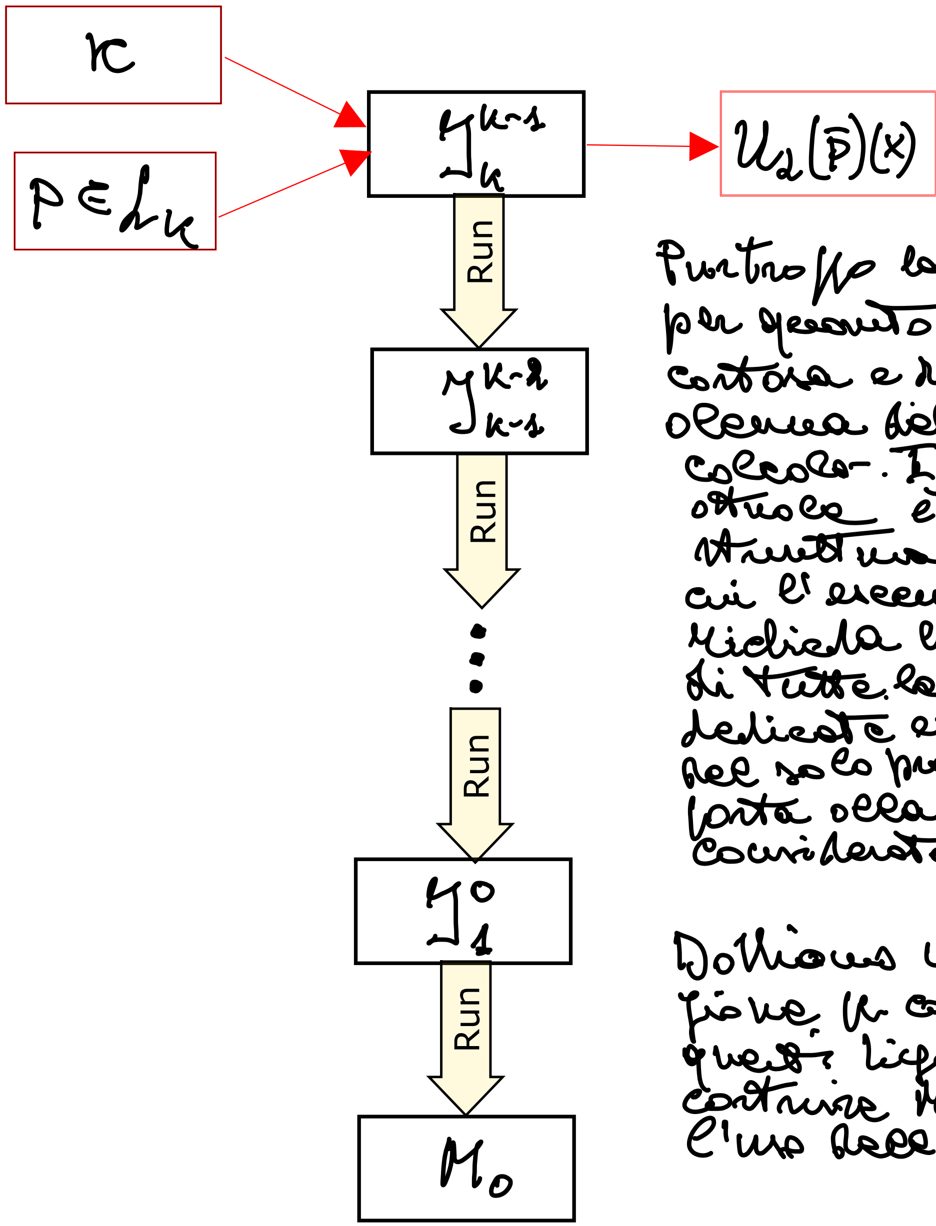


$$
\begin{tikzcd}
K \arrow[dr] & & \\
P \in L_K \arrow[r] & \mathcal{I}_K^{K-1} \arrow[r] \arrow[d, "\text{Run}"] & \mathcal{U}_L(\bar{P})(x) \\
& \mathcal{I}_{K-1}^{K-2} \arrow[d, "\text{Run}"] & \\
& \vdots \arrow[d, "\text{Run}"] & \\
& \mathcal{I}_1^0 \arrow[d, "\text{Run}"] & \\
& M_0 &
\end{tikzcd}
$$

Costruzione di esecutori per una cascata di linguaggi $L_K > L_{K-1} > \dots > L_0$ con L_0 avente M_0 concreto

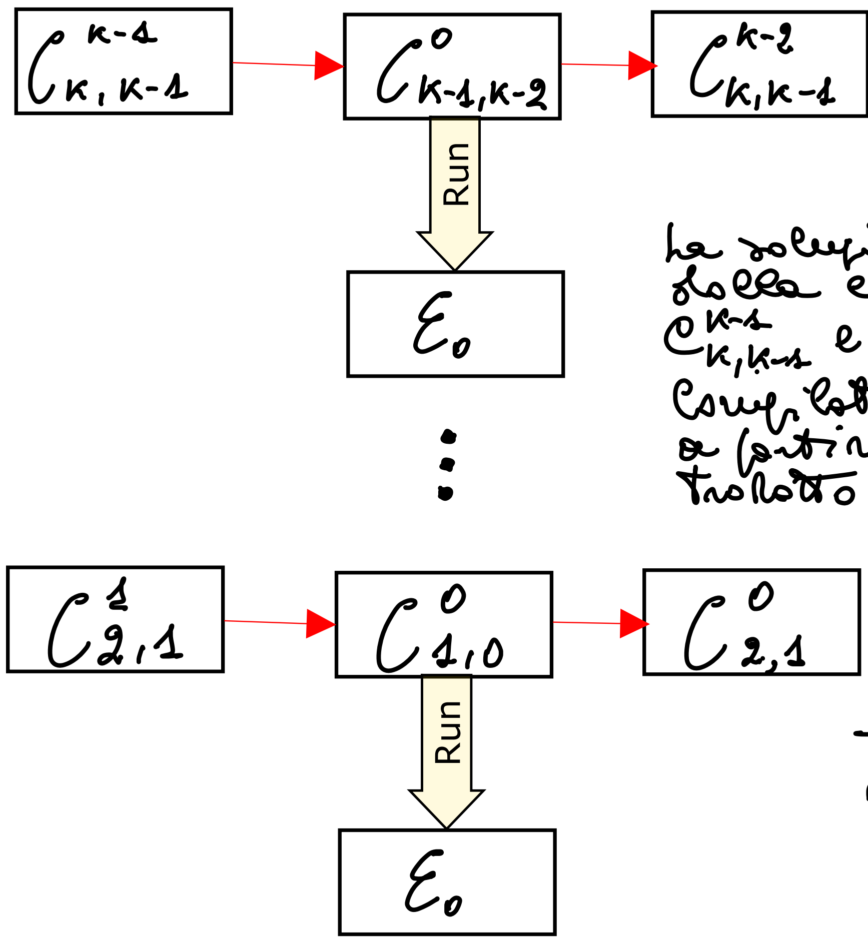
M_0 con esecutore E_0

Forniamo Interprete (e Compilatore) per ogni linguaggio L_i $(0 < i \leq K)$



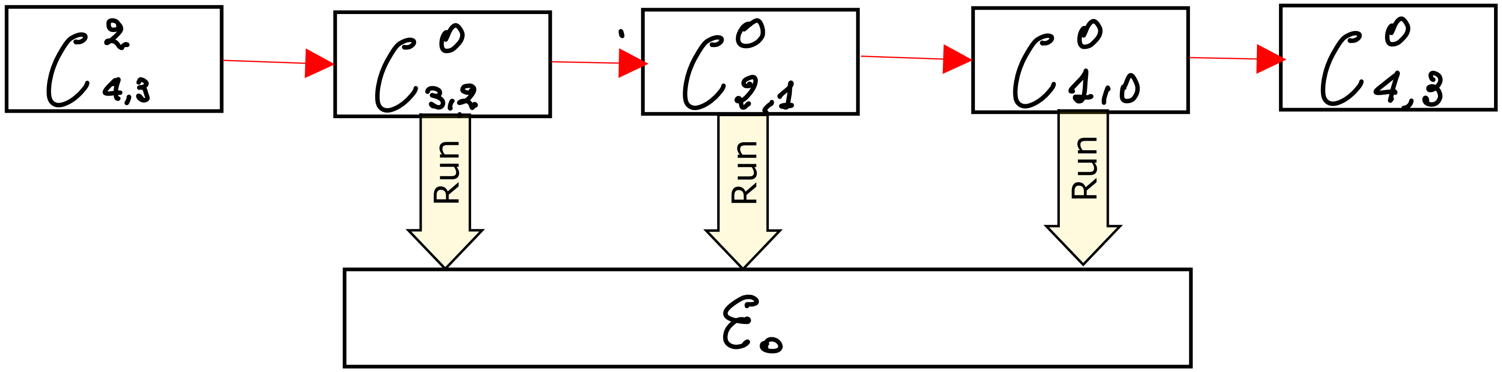
Purtroppo la costruzione a piramide per quanto corretta è estremamente costosa e di fatto non realizzabile su alcuna delle attuali piattaforme di calcolo. In particolare nessun S.O. attuale è in grado di supportare una struttura di macchine astratte in cui l'esecuzione di ogni macchina richieda la contemporanea esecuzione di tutte le macchine sottostanti: tutte dedicate esclusivamente all'esecuzione del solo programma, P, della macchina posta alla sommità della struttura considerata.

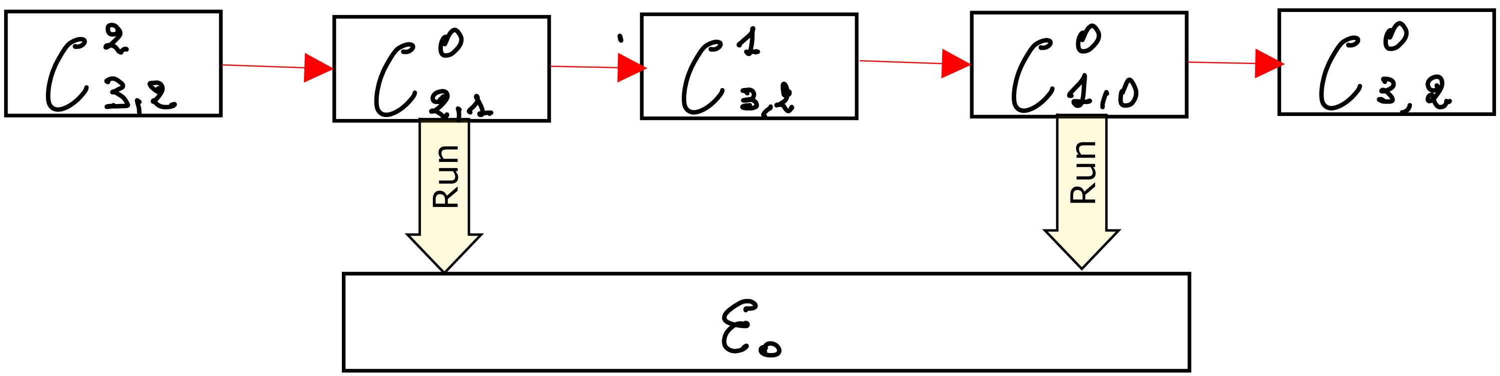
Dobbiamo trovare una diversa soluzione per costruire macchine per tutti questi linguaggi. La soluzione deve costruire macchine eseguibili senza l'uso dell'intera struttura.

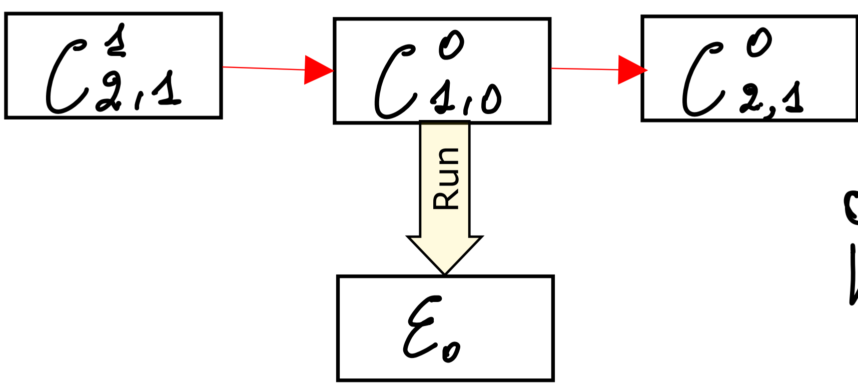


La soluzione che richiede parte della costruzione di compilatori $C^{k-1}_{k,k-1}$ e la loro compilazione su compilatori dell'intera catena a partire da quello di indice $k=2$ trovato in $C^{0}_{2,1}$ dal compilatore $C^{0}_{1,0}$.

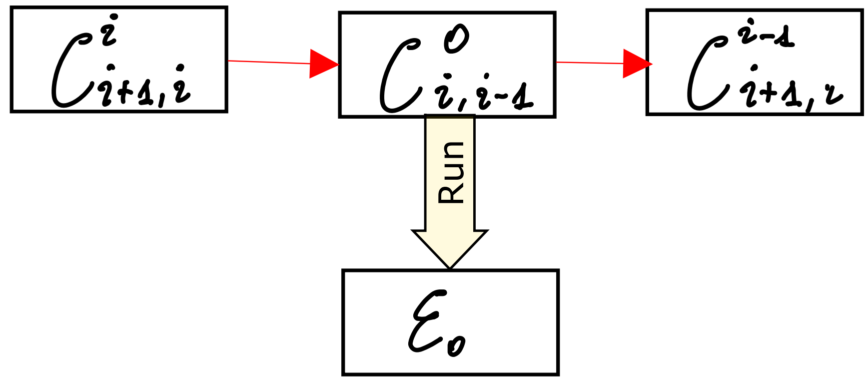
Questi compilatori sono facili da realizzare in quanto coinvolgono linguaggi di espressività prossima tra loro L_k e L_{k-1} e sono scritti nel linguaggio L_k più espressivo.

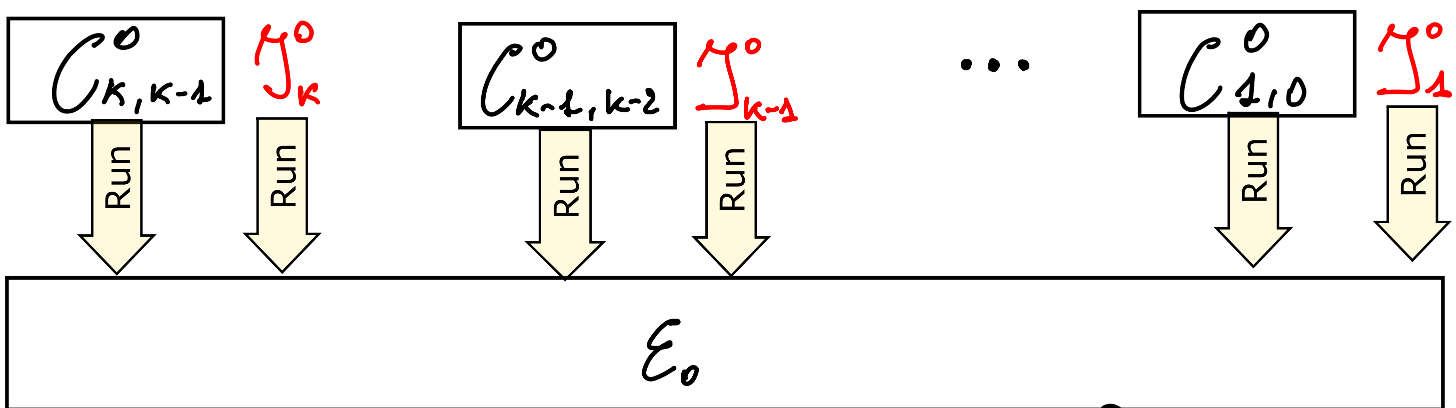






Qui vediamo i passi necessari per ottenere i compilatori di L_1, L_2, L_3 tutti eseguibili su M_0.





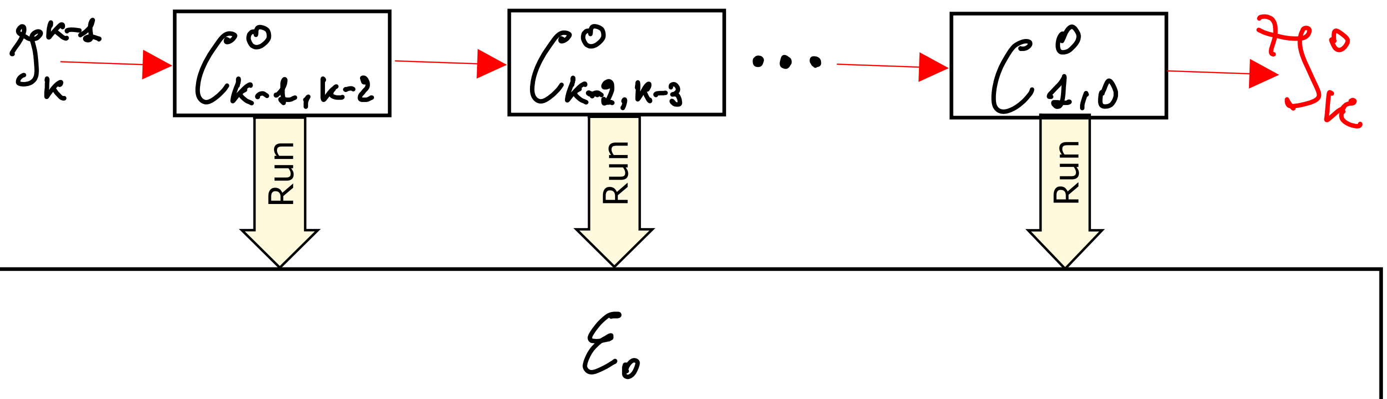
Una volta ottenuti i compilatori $C^0_{k,k-1}$ possiamo scrivere gli interpreti $\mathcal{I}^{k-1}_k$ ed ottenere $\mathcal{I}^0_k$ attraverso esecuzioni come questa sotto che esegue i compilatori in PIPE (vedi altra slide)

$$\mathcal{I}^0_k \equiv C^0_{k-1,k-2}\ \mathcal{I}^{k-1}_k \mid C^0_{k-2,k-3} \mid \dots \mid C^0_{1,0}$$

Il calcolo di $\mathcal{I}_K^0$ (e degli altri interpreti) può essere espresso con la formula sotto chiamata PIPE, in linux:

$$\mathcal{I}_K^0 = C^0_{K-1,K-2}\ \mathcal{I}_K^{K-1} \mid C^0_{K-2,K-3} \mid \dots \mid C^0_{1,0}$$

ed ha il seguente significato: Avvia l'esecuzione di $C^0_{K-1,K-2}$ con $\mathcal{I}_K^{K-1}$ come input, e al suo termine, di $C^0_{K-2,K-3} \dots C^0_{1,0}$, nell'ordine ciascuno con input l'output calcolato dal precedente al termine della propria computazione.



Anche i compilatori $C^0_{j,j-1}$ sono stati ottenuti con Pipe ed è la Pipe sotto che esprime in una formula il comportamento globalmente mostrato, 2 slide sopra, per il calcolo di $C^0_{4,3}$

$$C^0_{4,3} = C^0_{3,2}\ C^3_{4,3} \mid C^0_{2,1} \mid C^0_{1,0}$$