Linear Expansion

In this paper I will define the concept of linear expansion in the context of syntax parsing. We will progress through more and more complicated examples, beginning from the programming of a simple calculator until we ultimately have created an extensible programming language.

Table of Contents

1 Theoretical Background **2**

1.1 Stateless Expansion 2

1.2 Stated Expansion 3

1 Theoretical Background

1.1 Stateless Expansion

The idea of linear expansion is simple: given a string ξ the first character looks if it can bind with the second character to produce a new character, and the process repeats itself. There is of course, nuance. This nuance hides in the statement "if it can bind": we must define the rules for binding.

Let us define an *expander* to be a tuple (Σ, β) where Σ is an alphabet and $\beta \colon \overline{\Sigma} \times \overline{\Sigma} \hookrightarrow \overline{\Sigma}$ is a partial function called the *reduction function* where $\overline{\Sigma} = \Sigma \times \overline{\mathbb{N}}$, $\overline{\mathbb{N}} := \mathbb{N} \cup \{\infty\}$. A *program* over an expander is a string over $\overline{\Sigma}$. We write a program like $\sigma^1_{i_1} \cdots \sigma^n_{i_n}$ instead of as pairs $(\sigma^1, i_1) \dots (\sigma^n, i_n)$. In the character σ_i, we call i the *priority* of σ.

Then the rules of reduction are as follows, meaning we define $\beta(\xi)$ for a program: We do so in cases:

(1) If $\xi = \sigma_i$ then $\beta(\sigma) = \sigma_0$.

(2) If $\xi = \sigma^1_i \sigma^2_j \xi'$ where $i \geq j$ and $\beta(\sigma^1, \sigma^2) = \sigma^3_k$ is defined then $\beta(\xi) = \sigma^3_k \xi'$.

(3) Otherwise, $\beta(\xi) = \sigma^1_i \beta(\sigma^2_j \xi')$.

A string ξ such that $\beta(\xi) = \xi$ is called *irreducible*. Notice that it is possible for a string of length more than 1 to be irreducible: for example if $\beta(\sigma^1, \sigma^2)$ is not defined then $\sigma^1_i \sigma^2_j$ is irreducible.

$$\beta(\sigma_1 \tau_2) \xrightarrow{(3)} \sigma_1 \beta(\tau_2) \xrightarrow{(1)} \sigma_1 \tau_2$$

But such strings are not desired, since in the end we'd like a string to give us a value. So an irreducible string which is not a single character is called *ill-written*, and a string which is not ill-written is *well-written*.

Example: let $\Sigma = \mathbb{N} \cup \{+, \cdot\} \cup \{(n+), (n\cdot) \mid n \in \mathbb{N}\}$. β as follows:

σ^1_i, σ^2_j	$\beta(\sigma^1_i, \sigma^2_j)$
$n, +$	$(n+)$
$n, \cdot$	$(n\cdot)$
$(n+), m$	$n + m$
$(n\cdot), m$	$n \cdot m$
$(n\cdot), (m+)$	$(n \cdot m, +)$
$(n+), (m+)$	$(n + m, +)$
$(n\cdot), (m\cdot)$	$(n \cdot m, \cdot)$

Where n, m range over all values in $\mathbb{N}$. Here $\beta(\sigma_i, \sigma_j)$'s priority is j.

Now let us look at the string $1 + 2 \cdot 3 + 4;$. Here,

$$\begin{aligned}
1_\infty +_1 2_\infty \cdot_2 3_\infty +_1 4_\infty &\longrightarrow (1+)_1 2_\infty \cdot_2 3_\infty +_1 4_\infty \\
&\longrightarrow (1+)_1 (2\cdot)_2 3_\infty +_1 4_\infty \\
&\longrightarrow (1+)_1 (2\cdot)_2 (3+)_1 4_\infty \\
&\longrightarrow (1+)_1 (6+)_1 4_\infty \\
&\longrightarrow (7+)_1 4_\infty \\
&\longrightarrow (7+)_1 4_0 \\
&\longrightarrow (11)_0
\end{aligned}$$

So the rules for β we supplied seem to be sufficient for computing arithmetic expressions following the order of operations.

Example: We can also expand our language to include parentheses. So our alphabet becomes $\Sigma = \mathbb{N} \cup \{+, \cdot, (,)\} \cup \{\boldsymbol{(}n+\boldsymbol{)}, \boldsymbol{(}n\cdot\boldsymbol{)}, \boldsymbol{(}n\boldsymbol{)}) \mid n \in \mathbb{N}\}$. We distinguish between parentheses and bold parentheses for readability. We extend β as follows:

σ^1_i, σ^2_j	$\beta(\sigma^1_i, \sigma^2_j)$
$n, +$	$\boldsymbol{(}n+\boldsymbol{)}_j$
$n, \cdot$	$\boldsymbol{(}n\cdot\boldsymbol{)}_j$
$\boldsymbol{(}n+\boldsymbol{)}, m$	$(n + m)_j$
$\boldsymbol{(}n\cdot\boldsymbol{)}, m$	$(n \cdot m)_j$
$\boldsymbol{(}n\cdot\boldsymbol{)}, \boldsymbol{(}m+\boldsymbol{)}$	$\boldsymbol{(}n \cdot m, +\boldsymbol{)}_j$
$\boldsymbol{(}n+\boldsymbol{)}, \boldsymbol{(}m+\boldsymbol{)}$	$\boldsymbol{(}n + m, +\boldsymbol{)}_j$
$\boldsymbol{(}n\cdot\boldsymbol{)}, \boldsymbol{(}m\cdot\boldsymbol{)}$	$\boldsymbol{(}n \cdot m, \cdot\boldsymbol{)}_j$
$n,)$	$\boldsymbol{(}n\boldsymbol{)})_j$
$\boldsymbol{(}n+\boldsymbol{)}, \boldsymbol{(}m\boldsymbol{)})$	$\boldsymbol{(}n + m\boldsymbol{)})_j$
$\boldsymbol{(}n\cdot\boldsymbol{)}, \boldsymbol{(}m\boldsymbol{)})$	$\boldsymbol{(}n \cdot m\boldsymbol{)})_j$
$(, \boldsymbol{(}n\boldsymbol{)})$	n_i

$(n+m)_j$ means $n+m$ with a priority of j, not $\big(n+m\big)_j$. So for example expanding $2\cdot((1+2)\cdot 2)+1$,

$$\begin{aligned}
2_\infty *_2 (_\infty(_\infty 1_\infty +_1 2_\infty)_0 *_2 2_\infty)_0 +_1 1_\infty &\longrightarrow \big(2*\big)_2(_\infty(_\infty 1_\infty +_1 2_\infty)_0 *_2 2_\infty)_0 +_1 1_\infty\\
&\longrightarrow \big(2*\big)_2(_\infty(_\infty \big(1+\big)_1 2_\infty)_0 *_2 2_\infty)_0 +_1 1_\infty\\
&\longrightarrow \big(2*\big)_2(_\infty(_\infty \big(1+\big)_1 \big(2\big)\big)_0 *_2 2_\infty)_0 +_1 1_\infty\\
&\longrightarrow \big(2*\big)_2(_\infty(_\infty \big(3\big)\big)_0 *_2 2_\infty)_0 +_1 1_\infty\\
&\longrightarrow \big(2*\big)_2(_\infty 3_\infty *_2 2_\infty)_0 +_1 1_\infty\\
&\longrightarrow \big(2*\big)_2(_\infty \big(3*\big)_2 2_\infty)_0 +_1 1_\infty\\
&\longrightarrow \big(2*\big)_2(_\infty \big(3*\big)_2 \big(2\big)\big)_0 +_1 1_\infty\\
&\longrightarrow \big(2*\big)_2(_\infty \big(6\big)\big)_0 +_1 1_\infty\\
&\longrightarrow \big(2*\big)_2 6_\infty +_1 1_\infty\\
&\longrightarrow \big(2*\big)_2 \big(6+\big)_1 1_\infty\\
&\longrightarrow \big(12+\big)_1 1_\infty\\
&\longrightarrow \big(12+\big)_1 1_0\\
&\longrightarrow 13_0
\end{aligned}$$

Notice that in both examples, the characters in the string were given an initial priority. Thus we can extend our definition of an expander to include an *initial priority* function $\pi\colon\Sigma \hookrightarrow \overline{\mathbb{N}}$ which gives characters their initial priority. We can then canonically extend this to a function $\pi\colon\Sigma^* \hookrightarrow (\Sigma\times\overline{\mathbb{N}})^*$ defined by $\pi(\sigma^1\cdots\sigma^n) = \sigma^1_{\pi(\sigma^1)}\cdots\sigma^n_{\pi(\sigma^n)}$. Then a β-reduction of a string $\xi\in\Sigma^*$ is taken to mean a β-reduction of $\pi(\xi)$.

Notice that once again we require that π only be a partial function. This is since that we don't always need every character in Σ to have an initial priority; in the above examples there's no need to give $\big(n+\big)$ an inital priority, its priority is given through the β-reduction of n and $+$. So we now provide a new definition of a *program*, which is a string $\xi = \sigma^1\cdots\sigma^n\in\Sigma^*$ such that $\pi(\sigma^i)$ exists for all $1\le i\le n$. We can only of course discuss the reductions of programs, as $\pi(\xi)$ is only defined if ξ is a program.

1.2 Stated Expansion

Suppose we'd like to expand a program with variables in it. Then we cannot just use the previous definitions, as the actions of σ (which is to be understood as the function $\beta(\sigma,\bullet)$) are determined before any expansion occurs. We need a way to store the value of variables, a state.

This leads us to the following definition: let Σ_P and Σ_A be two disjoint sets of symbols: Σ_P the set of *printable symbols* and Σ_A the set of *abstract symbols*. Σ_P will generally be a set consisting of the string representations of abstract symbols, be it operators like $+$ and $\cdot$ or variable names. Σ_A are the actual objects which can "execute something". Let us further define $\Sigma = \Sigma_P \cupdot \Sigma_P$ and let $\mathsf{State} = \{\Sigma_P \longrightarrow \Sigma_A\cup\{\varepsilon\}\}$ be the set of all functions from printable to abstract symbols. We allow printable symbols to be mapped to ε, the empty word; this should be thought of the symbol having no value.

Now we begin with an initial β function which is a partial function

$$\beta\colon \overline{\Sigma}_A\times\overline{\Sigma}\times\mathsf{State}\hookrightarrow \big(\overline{\Sigma}\cupdot\{\varepsilon\}\big)\times\mathsf{State}$$

Recall that $\overline{\Sigma}$ is $\Sigma\times\mathbb{N}$. We will denote tuples in $X\times\mathsf{State}$ by $\langle x,s\rangle$ for $x\in X$ and $s\in\mathsf{State}$ for the sake of readability. So we now wish to extend to a β function

$$\beta\colon \overline{\Sigma}^*\times\mathsf{State}\hookrightarrow\overline{\Sigma}^*\times\mathsf{State}$$

We do this as follows: given $\xi\in(\Sigma\times\overline{\mathbb{N}})^*$ and $s\in\mathsf{State}$ we define $\beta\langle\xi,s\rangle$ as follows:

(1) if $\xi=\sigma_i\xi'$ for $\sigma\in\Sigma_P$ then $\beta\langle\xi,s\rangle = \langle s(\sigma)_i\xi',s\rangle$,

(2) if $\xi=\sigma^1_i\sigma^2_j\xi'$ for $\sigma^1\in\Sigma_A$, $i\ge j$, such that $\beta\big\langle\sigma^1_i\sigma^2_j,s\big\rangle = \big\langle\sigma^3_k,s'\big\rangle$ is defined, then $\beta\langle\xi,s\rangle = \big\langle\sigma^3_k\xi',s'\big\rangle$,

(3) otherwise if $\beta\big\langle\sigma^2_j\xi',s\big\rangle = \langle\xi'',s'\rangle$ then $\beta\langle\xi,s\rangle = \big\langle\sigma^1_i\xi'',s'\big\rangle$.

We also define the *initial priority function* to be a map $\pi\colon\Sigma_P\longrightarrow\overline{\mathbb{N}}$ (this is not a partial function: every printable symbol must be given a priority). This is once again canonically extended to a function $\pi\colon\Sigma_P^*\longrightarrow(\Sigma_P\times\overline{\mathbb{N}})^*$. And an *initial state* $s_0\in\mathsf{State}$. The quintuple $(\Sigma_P,\Sigma_A,\beta,\pi,s_0)$ is called an *expander*. The expansion of a string $\xi\in S$ is the process of iteratively applying β to $\langle\pi(\xi),s_0\rangle$.

Example: So for example let

$$\begin{aligned}
\Sigma_P &= \mathbb{N}\cup\{+,\cdot,=,;\}\cup\{\mathtt{let}\}\cup\big\{x^i \bigm| i\in\mathbb{N}\big\},\\
\Sigma_A &= \mathbb{N}\cup\{(n+),(n\cdot)\mid n\in\mathbb{N}\}\cup\{\mathtt{let}\}\cup\big\{(\mathtt{let}x^i),(\mathtt{let}x^i =)\bigm| i\in\mathbb{N}\big\}
\end{aligned}$$

where the natural numbers in Σ_A are not the same as the natural numbers in Σ_P since they must be disjoint, same for `let`. But they both essentially represent the same thing: s_0 maps $n \mapsto n$ for $n \in \mathbb{N}$ (the left-hand n is in Σ_p, the right-hand n is in Σ_A) and $\texttt{let} \mapsto \texttt{let}$. All other printable symbols are mapped to ε.

And similar to the previous example we define $\pi(n) = \infty$, $\pi(+) = 1$, and $\pi(\cdot) = 2$. We extend this to $\pi(;) = 0$, $\pi(=) = 0$, $\pi(\texttt{let}) = \infty$, and $\pi(x^i) = \infty$.

Let us take the same transitions as the example in the previous section for $n, (n+), (n\cdot)$ (we have to add the condition that the state doesnt change). We further add the transitions

$\langle \sigma^1\sigma^2, s \rangle$	$\beta\langle \sigma^1\sigma^2, s \rangle$
$\langle \sigma; , s \rangle$	$\langle \sigma_0, s \rangle$
$\langle \texttt{let}\ x^i, s \rangle$	$\langle (\texttt{let}\ x^i)_\infty, s \rangle$
$\langle (\texttt{let}\ x^i) =, s \rangle$	$\langle (\texttt{let}\ x^i =)_0, s \rangle$
$\langle (\texttt{let}\ x^i =)n, s \rangle$	$\langle \varepsilon, s[x^i \mapsto s(n)] \rangle$

In the final transition, $n \in \Sigma_A$. Then for example (we will be skipping trivial expansions):

$$\begin{aligned}
\texttt{let}\ x^1 = 1+2;\ \texttt{let}\ x^2 = 2;\ x^1 \cdot x^2; &\longrightarrow \texttt{let}_\infty x^1_\infty =_0 1_\infty +_1 2_\infty ;_0\ \texttt{let}_\infty x^2_\infty =_0 2_\infty ;_0\ x^1_\infty \cdot_2 x^2_\infty ;_0 \\
s_0 &\longrightarrow (\texttt{let} x^1 =)_0 1_\infty +_1 2_\infty ;_0\ \texttt{let}_\infty x^2_\infty =_0 2_\infty ;_0\ x^1_\infty \cdot_2 x^2_\infty ;_0 \\
s_0 &\longrightarrow (\texttt{let} x^1 =)_0 3_0\ \texttt{let}_\infty x^2_\infty =_0 2_\infty ;_0\ x^1_\infty \cdot_2 x^2_\infty ;_0 \\
s_0[x^1 \mapsto 3] &\longrightarrow \texttt{let}_\infty x^2_\infty =_0 2_\infty ;_0\ x^1_\infty \cdot_2 x^2_\infty ;_0 \\
s_0[x^1 \mapsto 3,\ x^2 \mapsto 2] &\longrightarrow x^1_\infty \cdot_2 x^2_\infty ;_0 \\
s_0[x^1 \mapsto 3,\ x^2 \mapsto 2] &\longrightarrow 3_\infty \cdot_2 x^2_\infty ;_0 \\
s_0[x^1 \mapsto 3,\ x^2 \mapsto 2] &\longrightarrow (3\cdot)_2 x^2_\infty ;_0 \\
s_0[x^1 \mapsto 3,\ x^2 \mapsto 2] &\longrightarrow (3\cdot)_2 2_\infty ;_0 \\
s_0[x^1 \mapsto 3,\ x^2 \mapsto 2] &\longrightarrow 6_0
\end{aligned}$$