Fragment Grammars

Maximilian Meffert

DISCLAIMER
Still needs proper citation!

Abstract. These notes summarize ideas on fragments of formal grammars and languages.

Consider the following code describing some sort of hierarchy:

```
Foo { Bar { Foo {} } }
```

we want to formally define all *fragments* (well-formed partial code pieces) of that code, i.e.:

```
Foo { Bar { Foo {} } }  // the code itself
Bar { Foo {} }          // the first piece
Foo {}                  // the second piece
Foo                     // a terminal
Bar                     // another terminal
{                       // yet another terminal
```

A *Formal Grammar* is a 4-tuple:

$$G := (V, \Sigma, P, S)$$

with:

- V a finite set of variables
- Σ a finite set of terminal symbols (the alphabet)
- $P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$ a finite set of production rules $x \to x'$
- $S \in V$ a start variable
- $V \cap \Sigma = \emptyset$ holds

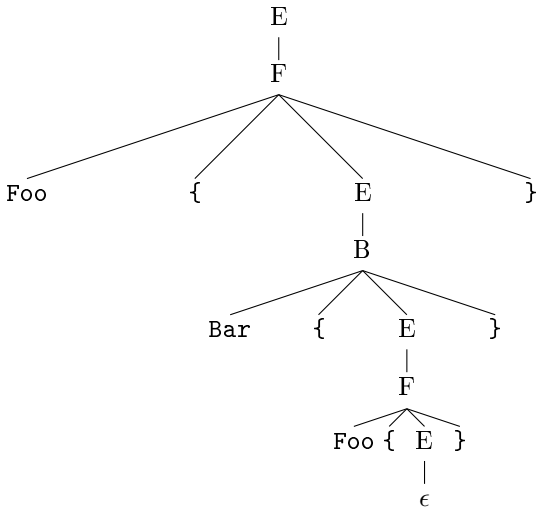
and the *Formal Language* generated by a grammar is the set:

$$\mathcal{L}(G) := \{w \in \Sigma^* | S \Rightarrow_G^* w\}$$

A possible grammar for the code above may be:

$$G = (\{E, F, B\}, \{\epsilon, \texttt{Foo}, \texttt{Bar}, \texttt{\{}, \texttt{\}}\}, P, E)$$
$$P = \{E \to \epsilon, E \to F, E \to B, F \to \texttt{Foo \{}E\texttt{\}}, B \to \texttt{Bar \{}E\texttt{\}}\}$$

producing the following syntax tree:



One can observe that each fragment is a derivation not necessarily starting with the initial start variable of the grammar. From this, we can also assume that each fragment is governed by its own grammar. For example, the fragments can be generated by grammars derived from the initial grammar just by switching the start variable:

- `Bar { Foo {} }`

  $G' = (V_G, \Sigma_G, P_G, B)$
- `Foo {}`

  $G'' = (V_G, \Sigma_G, P_G, F)$

So for any given grammar there is a set of grammars of which it is also an element:

$$G \in \bigcup_{v \in V_G} (V_G, \Sigma_G, P_G, v)$$

Note, that these fragment grammars are restricted to context-free grammars. Our method of switching start variables assumes $P \subseteq V \times (V \cup \Sigma)^*$.

To also address terminal symbols as fragments, we construct a set of minimal context-free grammars only capable of deriving terminals:

$$\bigcup_{t \in \Sigma_G} (\{V_t\}, \{t\}, \{V_t \to t\}, V_t)$$

For example:

$$G''' = (\{V_{\texttt{Foo}}\}, \{\texttt{Foo}\}, \{V_{\texttt{Foo}} \to \texttt{Foo}\}, V_{\texttt{Foo}})$$

Putting both sets together, we obtain the set of all *Fragment Grammar* for a given grammar:

$$F(G) = \bigcup_{v \in V_G} (V_G, \Sigma_G, P_G, v) \cup \bigcup_{t \in \Sigma_G} (\{V_t\}, \{t\}, \{V_t \to t\}, V_t)$$

Note, since context-free grammars are closed under union, $\mathcal{F}(G)$ is also context-free. For our example grammar above the has the fragment grammar:

$$\begin{aligned}
F(G) &= (V, \Sigma, P, S) \\
V &= \{E, F, B, V_\epsilon, V_{\texttt{Foo}}, V_{\texttt{Bar}}, V_{\texttt{\{}}, V_{\texttt{\}}}\} \\
\Sigma &= \{\epsilon, \texttt{Foo}, \texttt{Bar}, \texttt{\{}, \texttt{\}}\} \\
P &= \{E \to \epsilon, E \to F, E \to B, F \to \texttt{Foo \{}E\texttt{\}}, B \to \texttt{Bar \{}E\texttt{\}}, \\
&\quad V_\epsilon \to \epsilon, V_{\texttt{Foo}} \to \texttt{Foo}, V_{\texttt{Bar}} \to \texttt{Bar}, V_{\texttt{\{}} \to \texttt{\{}, V_{\texttt{\}}} \to \texttt{\}}, \\
&\quad S \to E, S \to F, S \to B, S \to V_\epsilon, S \to V_{\texttt{Foo}}, S \to V_{\texttt{Bar}}, S \to V_{\texttt{\{}}, S \to V_{\texttt{\}}}\}
\end{aligned}$$

Now we can also define a *Fragment Language* generated by a given grammar, containing all fragments which can be produced by that grammar:

$$\mathcal{F}(G) = \mathcal{L}(F(G)) = \bigcup_{g \in F(G)} \{w \in \Sigma^* | S_g \Rightarrow_g^* w\} = \bigcup_{g \in F(G)} \mathcal{L}(g)$$