Xtext Tutorial

February 2, 2024

1 Syntax Overview

In this chapter we show a syntax overview of Xtext with a meta model of a real-world example. Instead of using the generated Xtext code, we will rewrite the default grammar, generated by Xtext, into a simpler, more human-friendly one. A hands-on guide for using Xtext in Eclipse, with instructions on how to exactly generate the infrastructure and to run the generated editor is given in Chapter AD.

The Xtext specification language is a variation of the familiar Extended Backus-Naur Form (EBNF) notation for context free grammars. EBNF is used by most parser generators, and it is included in curriculum of most compiler courses.

We shall now discuss the main syntactic elements of the Xtext language, by presenting a simple grammar based on a tiny subset of the IUPAC nomenclature, whose meta-model is shown in Fig. 3.

„IUPAC“ is the abbreviation for „International Union of Pure and Applied Chemistry“ and is a nomenclature system, that is widely used in the chemistry. Main goal of this naming system is the encoding of chemical structures in a text-based name.

The two main components are chains and branches of carbon atoms. One part of the molecule is defined as chain and the rest are the branches. Every branch has the information on which carbon atom on the chain is connected to. The length information is simply substituted with a name. See table 1 for the substitution of chains and table 2.

Number of carbon atoms in branch	Encoded name
1	Methan
2	Ethan
3	Propan
4	Butan
5	Pentan
6	Hexan
7	Heptan
8	Octan
9	Nonan
10	Decan

Table 1: List of the first 10 names for a chain

For the encoded branch names the postfix „an“ of the encoded chain names is replaced with „yl“.

Number of carbon atoms in branch	Encoded name
1	Methyl
2	Ethyl
3	Propyl
4	Butyl

Table 2: List of the first 4 names for a branch

Usually if a branch with the same length occurs multiple times, these branches are summarized in one branch term and the number of branches with this length is also substituted with a string. The table 3 shows the first four elements. We call this element „summary prefix“.

Branches with the same length	Prefix
1	Mono
2	Di
3	Tri
4	Tetra

Table 3: List of the first 4 prefixes to summarize branches with the same length

In the following example the chain is red and the branch are drawn green .

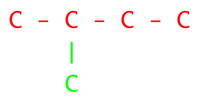

Figure 1: Chain length 4 → „Butan“; One branch with length 1 → „MonoMethyl“; Position of the branch: 2   Result: **2-MonoMethylButan**

The order of the components is: Position(s) + minus sign + summary prefix + encoded branch length + encoded chain length. In the case, that there is no branch, only the encoded chain length literal will be used. The next example shows, that this order is also valid for multiple different branches.

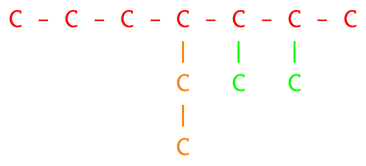

Figure 2: Chain length 7 → „Heptan“; Two branches with length 1 on position 2 and 3 → „2,3-DiMethyl“; One branch with length 2 on position 4 → „4-MonoEthyl“   Result: **2-MonoEthyl-3,4DiMethylButan**

Figure 3 shows a possible abstract syntax (the meta model) of the IUPAC nomenclature.

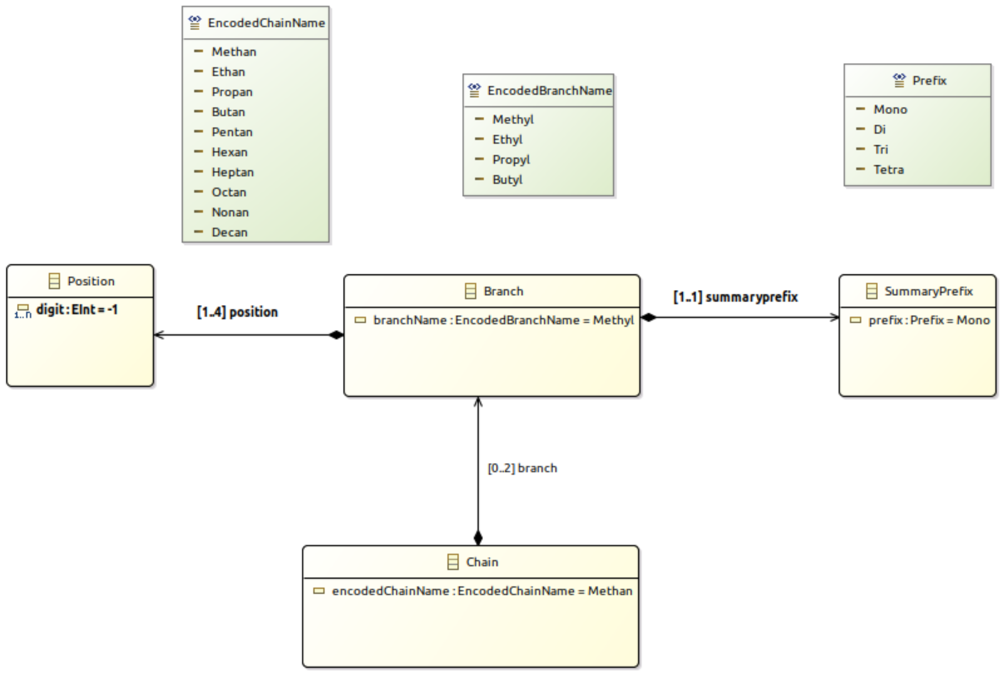

Figure 3: Meta model of the IUPAC nomenclature.

Terminals are simply introduced as string literals. Enums are in this context also kind of terminals. For the enum *Prefix*, the code look like:

```
enum Prefix returns Prefix:
        Mono = 'Mono' | Di = 'Di' | Tri = 'Tri' | Tetra = 'Tetra';
```

Here is important to notice, that the element left from the equal char is one of the enum entries and on the right side is the coressponding string literal. This double usage in Xtext sounds unnecessary, but the reason for this syntax descision is, that an enum entry could also have coressponding literals, that are different from the enum entry name.

Nonterminal symbols uses the property name, that was defined in the meta model. In addition the type of the property will be included here. The nonterminal „Position" could have the following implementation:

```
Position returns Position:
        digit=EInt ("," digit+=EInt)*;
```

We see here also the grouping operator and the repetition-symbol from EBNF. The += operator adds content to a property instead of assigning. A value resulting from parsing a nonterminal can be stored directly in a property of the current abstract syntax object. The meaning of the complete nonterminal: *A digit plus unlimited optional digits, that needs to be started with a comma char.*

Due to the composition relations between the classes, the class „Chain" is the class higest order and contains all other classes with direct oder indirect relations. We see in the description of the IUPAC nomenclature, that only a instance of the Chain class is mandatory. In Xtext such a mandatory can be expressed with braced char:

```
Chain returns Chain:
    {Chain}
        (branch+=Branch (branch+=Branch)*)?
        (encodedChainName=EncodedChainName);
```

Don't be confused with the repetition meaning in EBNF! In Xtext a repetition can only be written with the char * and +. The question mark has the same meaning compared to EBNF; it describes that an terminal or nonterminal is optional.

Vergleichstabelle Xtext EBNF
Kompletten Xtext Code hier anzeigen.

There is much more to Xtext than we present, but this is sufficient for creating simple languages. Among other elements, of the highest interest are probably customizable scoping semantics (what names are visible in what scopes), and fully qualified name support for references across name spaces/scopes. These are described in the Xtext documentation.

Finally, C-like comments (both block comments, and line comments) are automatically supported in languages built with Xtext.