Sample Extended BNF: 
Haskell expression fragment

This BNF for Haskell expressions is taken directly from the Haskell 98 Report, but extracts only selected parts.

```
exp □  exp0
expi □  expi+1 [qop(n,i) expi+1] □ lexp i □ rexp i  
           (0≤ i ≤9)
lexp □  (lexp i expi+1) qop(l,i) expi+1  
           (0≤ i ≤9, left-assoc)
rexp □  expi+1 qop(r,i) (rexp i expi+1)  
           (0≤ i ≤9, right-assoc)
qop(n,4) □ < | <= | == | /= | >= | >  
           (non-associative, prec 4)
qop(l,6) □ + | –  
           (left-associative, prec 6)
qop(l,7) □ * | /  
           (left-associative, prec 7)
qop(r,8) □ ** | ^^ | ^  
           (right-associative, prec 8)
qop(r,9) □ .  
           (right-associative, prec 9)
exp10 □ \ apat1 ... apatn -> exp  
           (lambda abstraction, n≥1)
    | let decls in exp  
    | if exp then exp else exp  
    | case exp of { alts }  
    | do { stmts }  
    | fexp
fexp □ [ fexp ] aexp
           (optional function application)
aexp □ qvar
    | gcon
    | literal
    | ( exp )
    | (exp1, ..., expn)
    | [exp1, ..., expn]
qvar □ identifier
literal □ numeral | charconst | stringconst | boolconst
```

partial EBNF for Haskell expressions

Operations and expressions are categorized as left/right/non-associative, and the syntax categories provided reflect both this and the precedence of operations. A
more complete table of Haskell operators appears below. Haskell syntax for variables and numbers is not included here, but is not significantly different from other programming languages. Note that a number of symbols (e.g., \([, \]) are used both as Haskell characters and BNF markup, and the Haskell BNF distinguishes their use by different fonts — this is a serious potential source of confusion. The BNF instances of these symbols are written in bold in the BNF above.

The Haskell operators, their associativity and precedence, are summarized in the following table:

<table>
<thead>
<tr>
<th>prec</th>
<th>left-assoc</th>
<th>non-assoc</th>
<th>right-assoc</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>!!</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>**, ^^, ^</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>*, /, %, <code>div', </code>mod', <code>rem', </code>quot'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>+, -</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>:, ++</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/=, &lt;, &lt;=, ==, &gt;, &gt;=, <code>elem', </code>notElem'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td><code>&amp;</code></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td><code>|</code></td>
</tr>
<tr>
<td>1</td>
<td>&lt;=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>$, `seq'</td>
</tr>
</tbody>
</table>