The Language GF

BNF-converter

November 8, 2004

This document was automatically generated by the *BNF-Converter*. It was generated together with the lexer, the parser, and the abstract syntax module, which guarantees that the document matches with the implementation of the language (provided no hand-hacking has taken place).

The lexical structure of GF

Identifiers

Identifiers ⟨*Ident*⟩ are unquoted strings beginning with a letter, followed by any combination of letters, digits, and the characters `_ '`, reserved words excluded.

Literals

String literals ⟨*String*⟩ have the form `"`x`"`, where x is any sequence of any characters except `"` unless preceded by `\`.

Integer literals ⟨*Int*⟩ are nonempty sequences of digits.

LString literals are recognized by the regular expression ‘”(⟨*anychar*⟩ − ‘”)∗‘”

Reserved words and symbols

The set of reserved words is the set of terminals appearing in the grammar. Those reserved words that consist of non-letter characters are called symbols, and they are treated in a different way from those that are similar to identifiers. The lexer follows rules familiar from languages like Haskell, C, and Java, including longest match and spacing conventions.

The reserved words used in GF are the following:

Lin	PType	Str
Strs	Tok	Type
abstract	case	cat
concrete	data	def
flags	fn	fun
grammar	in	include
incomplete	instance	interface
let	lin	lincat
lindef	lintype	of
open	oper	out
package	param	pattern
pre	printname	resource
reuse	strs	table
tokenizer	transfer	union
var	variants	where
with		

The symbols used in GF are the following:

;	=	{
}	(	)
:	−>	**
,	[	]
.	|	?
<	>	@
!	*	\
=>	++	+
_	$	/
−		

Comments

Single-line comments begin with −−.
Multiple-line comments are enclosed with {− and −}.

The syntactic structure of GF

Non-terminals are enclosed between ⟨ and ⟩. The symbols ::= (production), | (union) and ϵ (empty rule) belong to the BNF notation. All other symbols are terminals.

$$\langle \mathit{Grammar} \rangle \quad ::= \quad \langle \mathit{ListModDef} \rangle$$

```
⟨ListModDef⟩    ::=  ε
                 |   ⟨ModDef⟩ ⟨ListModDef⟩

⟨ModDef⟩    ::=  ⟨ModDef⟩ ;
             |   grammar ⟨Ident⟩ = { abstract = ⟨Ident⟩ ; ⟨ListConcSpec⟩ }
             |   ⟨ComplMod⟩ ⟨ModType⟩ = ⟨ModBody⟩

⟨ConcSpec⟩    ::=  ⟨Ident⟩ = ⟨ConcExp⟩

⟨ListConcSpec⟩    ::=  ε
                   |   ⟨ConcSpec⟩
                   |   ⟨ConcSpec⟩ ; ⟨ListConcSpec⟩

⟨ConcExp⟩    ::=  ⟨Ident⟩ ⟨ListTransfer⟩

⟨ListTransfer⟩    ::=  ε
                   |   ⟨Transfer⟩ ⟨ListTransfer⟩

⟨Transfer⟩    ::=  ( transfer in ⟨Open⟩ )
               |   ( transfer out ⟨Open⟩ )

⟨ModType⟩    ::=  abstract ⟨Ident⟩
              |   resource ⟨Ident⟩
              |   interface ⟨Ident⟩
              |   concrete ⟨Ident⟩ of ⟨Ident⟩
              |   instance ⟨Ident⟩ of ⟨Ident⟩
              |   transfer ⟨Ident⟩ : ⟨Open⟩ -> ⟨Open⟩

⟨ModBody⟩    ::=  ⟨Extend⟩ ⟨Opens⟩ { ⟨ListTopDef⟩ }
              |   ⟨Ident⟩ with ⟨ListOpen⟩
              |   reuse ⟨Ident⟩
              |   union ⟨ListIncluded⟩

⟨ListTopDef⟩    ::=  ε
                 |   ⟨TopDef⟩ ⟨ListTopDef⟩

⟨Extend⟩    ::=  ⟨ListIdent⟩ **
             |   ε

⟨ListOpen⟩    ::=  ε
               |   ⟨Open⟩
               |   ⟨Open⟩ , ⟨ListOpen⟩

⟨Opens⟩    ::=  ε
            |   open ⟨ListOpen⟩ in

⟨Open⟩    ::=  ⟨Ident⟩
           |   ( ⟨QualOpen⟩ ⟨Ident⟩ )
           |   ( ⟨QualOpen⟩ ⟨Ident⟩ = ⟨Ident⟩ )
```

⟨ComplMod⟩ ::= ϵ
  | `incomplete`

⟨QualOpen⟩ ::= ϵ
  | `incomplete`
  | `interface`

⟨ListIncluded⟩ ::= ϵ
  | *⟨Included⟩*
  | *⟨Included⟩* `,` *⟨ListIncluded⟩*

⟨Included⟩ ::= *⟨Ident⟩*
  | *⟨Ident⟩* `[` *⟨ListIdent⟩* `]`

⟨Def⟩ ::= *⟨ListIdent⟩* `:` *⟨Exp⟩*
  | *⟨ListIdent⟩* = *⟨Exp⟩*
  | *⟨Ident⟩* *⟨ListPatt⟩* = *⟨Exp⟩*
  | *⟨ListIdent⟩* `:` *⟨Exp⟩* = *⟨Exp⟩*

⟨TopDef⟩ ::= `cat` *⟨ListCatDef⟩*
  | `fun` *⟨ListFunDef⟩*
  | `data` *⟨ListFunDef⟩*
  | `def` *⟨ListDef⟩*
  | `data` *⟨ListDataDef⟩*
  | `transfer` *⟨ListDef⟩*
  | `param` *⟨ListParDef⟩*
  | `oper` *⟨ListDef⟩*
  | `lincat` *⟨ListPrintDef⟩*
  | `lindef` *⟨ListDef⟩*
  | `lin` *⟨ListDef⟩*
  | `printname cat` *⟨ListPrintDef⟩*
  | `printname fun` *⟨ListPrintDef⟩*
  | `flags` *⟨ListFlagDef⟩*
  | `printname` *⟨ListPrintDef⟩*
  | `lintype` *⟨ListDef⟩*
  | `pattern` *⟨ListDef⟩*
  | `package` *⟨Ident⟩* = { *⟨ListTopDef⟩* } `;`
  | `var` *⟨ListDef⟩*
  | `tokenizer` *⟨Ident⟩* `;`

⟨CatDef⟩ ::= *⟨Ident⟩* *⟨ListDDecl⟩*

⟨FunDef⟩ ::= *⟨ListIdent⟩* `:` *⟨Exp⟩*

⟨DataDef⟩ ::= *⟨Ident⟩* = *⟨ListDataConstr⟩*

$$\begin{array}{lll} \langle \mathit{DataConstr} \rangle & ::= & \langle \mathit{Ident} \rangle \\ & | & \langle \mathit{Ident} \rangle \ \texttt{.} \ \langle \mathit{Ident} \rangle \end{array}$$

$$\begin{array}{lll} \langle \mathit{ListDataConstr} \rangle & ::= & \epsilon \\ & | & \langle \mathit{DataConstr} \rangle \\ & | & \langle \mathit{DataConstr} \rangle \ | \ \langle \mathit{ListDataConstr} \rangle \end{array}$$

$$\begin{array}{lll} \langle \mathit{ParDef} \rangle & ::= & \langle \mathit{Ident} \rangle = \langle \mathit{ListParConstr} \rangle \\ & | & \langle \mathit{Ident} \rangle = \texttt{(} \ \texttt{in} \ \langle \mathit{Ident} \rangle \ \texttt{)} \\ & | & \langle \mathit{Ident} \rangle \end{array}$$

$$\begin{array}{lll} \langle \mathit{ParConstr} \rangle & ::= & \langle \mathit{Ident} \rangle \ \langle \mathit{ListDDecl} \rangle \end{array}$$

$$\begin{array}{lll} \langle \mathit{PrintDef} \rangle & ::= & \langle \mathit{ListIdent} \rangle = \langle \mathit{Exp} \rangle \end{array}$$

$$\begin{array}{lll} \langle \mathit{FlagDef} \rangle & ::= & \langle \mathit{Ident} \rangle = \langle \mathit{Ident} \rangle \end{array}$$

$$\begin{array}{lll} \langle \mathit{ListDef} \rangle & ::= & \langle \mathit{Def} \rangle \ \texttt{;} \\ & | & \langle \mathit{Def} \rangle \ \texttt{;} \ \langle \mathit{ListDef} \rangle \end{array}$$

$$\begin{array}{lll} \langle \mathit{ListCatDef} \rangle & ::= & \langle \mathit{CatDef} \rangle \ \texttt{;} \\ & | & \langle \mathit{CatDef} \rangle \ \texttt{;} \ \langle \mathit{ListCatDef} \rangle \end{array}$$

$$\begin{array}{lll} \langle \mathit{ListFunDef} \rangle & ::= & \langle \mathit{FunDef} \rangle \ \texttt{;} \\ & | & \langle \mathit{FunDef} \rangle \ \texttt{;} \ \langle \mathit{ListFunDef} \rangle \end{array}$$

$$\begin{array}{lll} \langle \mathit{ListDataDef} \rangle & ::= & \langle \mathit{DataDef} \rangle \ \texttt{;} \\ & | & \langle \mathit{DataDef} \rangle \ \texttt{;} \ \langle \mathit{ListDataDef} \rangle \end{array}$$

$$\begin{array}{lll} \langle \mathit{ListParDef} \rangle & ::= & \langle \mathit{ParDef} \rangle \ \texttt{;} \\ & | & \langle \mathit{ParDef} \rangle \ \texttt{;} \ \langle \mathit{ListParDef} \rangle \end{array}$$

$$\begin{array}{lll} \langle \mathit{ListPrintDef} \rangle & ::= & \langle \mathit{PrintDef} \rangle \ \texttt{;} \\ & | & \langle \mathit{PrintDef} \rangle \ \texttt{;} \ \langle \mathit{ListPrintDef} \rangle \end{array}$$

$$\begin{array}{lll} \langle \mathit{ListFlagDef} \rangle & ::= & \langle \mathit{FlagDef} \rangle \ \texttt{;} \\ & | & \langle \mathit{FlagDef} \rangle \ \texttt{;} \ \langle \mathit{ListFlagDef} \rangle \end{array}$$

$$\begin{array}{lll} \langle \mathit{ListParConstr} \rangle & ::= & \epsilon \\ & | & \langle \mathit{ParConstr} \rangle \\ & | & \langle \mathit{ParConstr} \rangle \ | \ \langle \mathit{ListParConstr} \rangle \end{array}$$

$$\begin{array}{lll} \langle \mathit{ListIdent} \rangle & ::= & \langle \mathit{Ident} \rangle \\ & | & \langle \mathit{Ident} \rangle \ \texttt{,} \ \langle \mathit{ListIdent} \rangle \end{array}$$

$$\begin{array}{lll} \langle \mathit{LocDef} \rangle & ::= & \langle \mathit{ListIdent} \rangle \ \texttt{:} \ \langle \mathit{Exp} \rangle \\ & | & \langle \mathit{ListIdent} \rangle = \langle \mathit{Exp} \rangle \\ & | & \langle \mathit{ListIdent} \rangle \ \texttt{:} \ \langle \mathit{Exp} \rangle = \langle \mathit{Exp} \rangle \end{array}$$

```
⟨ListLocDef⟩ ::= ϵ
               |  ⟨LocDef⟩
               |  ⟨LocDef⟩ ; ⟨ListLocDef⟩

⟨Exp4⟩ ::= ⟨Ident⟩
         |  { ⟨Ident⟩ }
         |  [ ⟨Ident⟩ ]
         |  ⟨Sort⟩
         |  ⟨String⟩
         |  ⟨Integer⟩
         |  ?
         |  [ ]
         |  data
         |  [ ⟨String⟩ ]
         |  { ⟨ListLocDef⟩ }
         |  < ⟨ListTupleComp⟩ >
         |  ( in ⟨Ident⟩ )
         |  < ⟨Exp⟩ : ⟨Exp⟩ >
         |  ( ⟨Exp⟩ )
         |  ⟨LString⟩

⟨Exp3⟩ ::= ⟨Exp3⟩ . ⟨Label⟩
         |  { ⟨Ident⟩ . ⟨Ident⟩ }
         |  [ ⟨Ident⟩ . ⟨Ident⟩ ]
         |  ⟨Exp4⟩

⟨Exp2⟩ ::= ⟨Exp2⟩ ⟨Exp3⟩
         |  table { ⟨ListCase⟩ }
         |  table ⟨Exp4⟩ { ⟨ListCase⟩ }
         |  case ⟨Exp⟩ of { ⟨ListCase⟩ }
         |  variants { ⟨ListExp⟩ }
         |  pre { ⟨Exp⟩ ; ⟨ListAltern⟩ }
         |  strs { ⟨ListExp⟩ }
         |  ⟨Ident⟩ @ ⟨Exp4⟩
         |  ⟨Exp3⟩
         |  Lin ⟨Ident⟩

⟨Exp1⟩ ::= ⟨Exp1⟩ ! ⟨Exp2⟩
         |  ⟨Exp1⟩ * ⟨Exp2⟩
         |  ⟨Exp1⟩ ** ⟨Exp2⟩
         |  ⟨Exp2⟩
```

$$
\begin{array}{lcl}
\langle \mathit{Exp} \rangle & ::= & \backslash\ \langle \mathit{ListBind} \rangle \ \texttt{->}\ \langle \mathit{Exp} \rangle \\
 & | & \backslash\ \backslash\ \langle \mathit{ListBind} \rangle \ \texttt{=>}\ \langle \mathit{Exp} \rangle \\
 & | & \langle \mathit{Decl} \rangle \ \texttt{->}\ \langle \mathit{Exp} \rangle \\
 & | & \langle \mathit{Exp1} \rangle \ \texttt{=>}\ \langle \mathit{Exp} \rangle \\
 & | & \langle \mathit{Exp1} \rangle \ \texttt{++}\ \langle \mathit{Exp} \rangle \\
 & | & \langle \mathit{Exp1} \rangle \ \texttt{+}\ \langle \mathit{Exp} \rangle \\
 & | & \texttt{let}\ \{\ \langle \mathit{ListLocDef} \rangle\ \}\ \texttt{in}\ \langle \mathit{Exp} \rangle \\
 & | & \texttt{let}\ \langle \mathit{ListLocDef} \rangle\ \texttt{in}\ \langle \mathit{Exp} \rangle \\
 & | & \langle \mathit{Exp1} \rangle\ \texttt{where}\ \{\ \langle \mathit{ListLocDef} \rangle\ \} \\
 & | & \texttt{fn}\ \{\ \langle \mathit{ListEquation} \rangle\ \} \\
 & | & \langle \mathit{Exp1} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{ListExp} \rangle & ::= & \epsilon \\
 & | & \langle \mathit{Exp} \rangle \\
 & | & \langle \mathit{Exp} \rangle\ \texttt{;}\ \langle \mathit{ListExp} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{Patt1} \rangle & ::= & \texttt{\_} \\
 & | & \langle \mathit{Ident} \rangle \\
 & | & \{\ \langle \mathit{Ident} \rangle\ \} \\
 & | & \langle \mathit{Ident} \rangle\ \texttt{.}\ \langle \mathit{Ident} \rangle \\
 & | & \langle \mathit{Integer} \rangle \\
 & | & \langle \mathit{String} \rangle \\
 & | & \{\ \langle \mathit{ListPattAss} \rangle\ \} \\
 & | & <\ \langle \mathit{ListPattTupleComp} \rangle\ > \\
 & | & \texttt{(}\ \langle \mathit{Patt} \rangle\ \texttt{)}
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{Patt} \rangle & ::= & \langle \mathit{Ident} \rangle\ \langle \mathit{ListPatt} \rangle \\
 & | & \langle \mathit{Ident} \rangle\ \texttt{.}\ \langle \mathit{Ident} \rangle\ \langle \mathit{ListPatt} \rangle \\
 & | & \langle \mathit{Patt1} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{PattAss} \rangle & ::= & \langle \mathit{ListIdent} \rangle\ =\ \langle \mathit{Patt} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{Label} \rangle & ::= & \langle \mathit{Ident} \rangle \\
 & | & \texttt{\$}\ \langle \mathit{Integer} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{Sort} \rangle & ::= & \texttt{Type} \\
 & | & \texttt{PType} \\
 & | & \texttt{Tok} \\
 & | & \texttt{Str} \\
 & | & \texttt{Strs}
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{ListPattAss} \rangle & ::= & \epsilon \\
 & | & \langle \mathit{PattAss} \rangle \\
 & | & \langle \mathit{PattAss} \rangle\ \texttt{;}\ \langle \mathit{ListPattAss} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{PattAlt} \rangle & ::= & \langle \mathit{Patt} \rangle
\end{array}
$$

```
⟨ListPatt⟩  ::=  ⟨Patt1⟩
             |   ⟨Patt1⟩ ⟨ListPatt⟩

⟨ListPattAlt⟩  ::=  ⟨PattAlt⟩
                |   ⟨PattAlt⟩ | ⟨ListPattAlt⟩

⟨Bind⟩  ::=  ⟨Ident⟩
         |   _

⟨ListBind⟩  ::=  ϵ
             |   ⟨Bind⟩
             |   ⟨Bind⟩ , ⟨ListBind⟩

⟨Decl⟩  ::=  ( ⟨ListBind⟩ : ⟨Exp⟩ )
         |   ⟨Exp2⟩

⟨TupleComp⟩  ::=  ⟨Exp⟩

⟨PattTupleComp⟩  ::=  ⟨Patt⟩

⟨ListTupleComp⟩  ::=  ϵ
                  |   ⟨TupleComp⟩
                  |   ⟨TupleComp⟩ , ⟨ListTupleComp⟩

⟨ListPattTupleComp⟩  ::=  ϵ
                      |   ⟨PattTupleComp⟩
                      |   ⟨PattTupleComp⟩ , ⟨ListPattTupleComp⟩

⟨Case⟩  ::=  ⟨ListPattAlt⟩ => ⟨Exp⟩

⟨ListCase⟩  ::=  ⟨Case⟩
             |   ⟨Case⟩ ; ⟨ListCase⟩

⟨Equation⟩  ::=  ⟨ListPatt⟩ -> ⟨Exp⟩

⟨ListEquation⟩  ::=  ϵ
                 |   ⟨Equation⟩
                 |   ⟨Equation⟩ ; ⟨ListEquation⟩

⟨Altern⟩  ::=  ⟨Exp⟩ / ⟨Exp⟩

⟨ListAltern⟩  ::=  ϵ
               |   ⟨Altern⟩
               |   ⟨Altern⟩ ; ⟨ListAltern⟩

⟨DDecl⟩  ::=  ( ⟨ListBind⟩ : ⟨Exp⟩ )
          |   ⟨Exp4⟩
```

$$\begin{array}{lcl}
\langle \mathit{ListDDecl} \rangle & ::= & \epsilon \\
 & | & \langle \mathit{DDecl} \rangle \ \langle \mathit{ListDDecl} \rangle
\end{array}$$

$$\begin{array}{lcl}
\langle \mathit{OldGrammar} \rangle & ::= & \langle \mathit{Include} \rangle \ \langle \mathit{ListTopDef} \rangle
\end{array}$$

$$\begin{array}{lcl}
\langle \mathit{Include} \rangle & ::= & \epsilon \\
 & | & \texttt{include} \ \langle \mathit{ListFileName} \rangle
\end{array}$$

$$\begin{array}{lcl}
\langle \mathit{FileName} \rangle & ::= & \langle \mathit{String} \rangle \\
 & | & \langle \mathit{Ident} \rangle \\
 & | & \texttt{/} \ \langle \mathit{FileName} \rangle \\
 & | & \texttt{.} \ \langle \mathit{FileName} \rangle \\
 & | & - \ \langle \mathit{FileName} \rangle \\
 & | & \langle \mathit{Ident} \rangle \ \langle \mathit{FileName} \rangle
\end{array}$$

$$\begin{array}{lcl}
\langle \mathit{ListFileName} \rangle & ::= & \langle \mathit{FileName} \rangle \ \texttt{;} \\
 & | & \langle \mathit{FileName} \rangle \ \texttt{;} \ \langle \mathit{ListFileName} \rangle
\end{array}$$