QCIR-G14: A Non-Prenex Non-CNF Format for Quantified Boolean Formulas

QBF Gallery 2014
http://qbf.satisfiability.org/gallery/

April 8, 2014

Contents

Acknowledgements

This document is based on an earlier version of the QCIR standard which was initiated and kindly provided by Alexandra Goultiaeva and Will Klieber. Extensive feedback was provided by Lukasz Kaiser. Further, many comments given by the participants of the QBF Workshop 2013 are included.

1 Introduction

This document defines the input format QCIRfor tools processing or producing *quantified Boolean formulas (QBF)*. The QCIRformat is based on the ISCAS-89 format. QCIRallows the representation of quantified circuits in prenex as well as in non-prenex form. The QCIRformat is designed for being easy to use in applications on the one hand and for being easy to be implemented in solvers and related tools on the other hand. In order to satisfy both requirements, the standard defines a general version providing much freedom to the user and a version defining cleansed formulas which are easier to process. This document first gives a concise definition of the structure of a QCIRformula followed by a textual description of implementation details which cannot be covered in terms of a grammar. Then restrictions to the cleansed format are introduced and examples are provided. Finally, this document concludes with a list of features to be included in the future.

2 Format Specification

2.1 Syntax

The following BNF grammar specifies the structure of a formula represented in QCIR (Quantified CIRcuit).

$$
\begin{array}{rcl}
\textit{qcir-file} & ::= & \textit{format-id qblock-stmt output-stmt (gate-stmt nl)}^* \\
\textit{format-id} & ::= & \texttt{\#QCIR-G14}\ [\textit{integer}]\ \textit{nl} \\
\textit{qblock-stmt} & ::= & [\texttt{free(}\textit{var-list}\texttt{)}\textit{nl}]\ \textit{qblock-quant}^* \\
\textit{qblock-quant} & ::= & \textit{quant}\texttt{(}\textit{var-list}\texttt{)}\textit{nl} \\
\textit{var-list} & ::= & \textit{(var}\texttt{,}\textit{)}^*\ \textit{var} \\
\textit{lit-list} & ::= & \textit{(lit}\texttt{,}\textit{)}^*\ \textit{lit} \mid \epsilon \\
\textit{output-stmt} & ::= & \texttt{output(}\textit{lit}\texttt{)}\textit{nl} \\
\textit{gate-stmt} & ::= & \textit{gvar}\ \texttt{=}\ \textit{ngate\_type}\texttt{(}\textit{lit-list}\texttt{)} \\
 & \mid & \textit{gvar}\ \texttt{=}\ \texttt{xor(}\textit{lit}\texttt{,}\ \textit{lit}\texttt{)} \\
 & \mid & \textit{gvar}\ \texttt{=}\ \texttt{ite(}\textit{lit}\texttt{,}\ \textit{lit}\texttt{,}\ \textit{lit}\texttt{)} \\
 & \mid & \textit{gvar}\ \texttt{=}\ \textit{quant}\texttt{(}\textit{var-list}\texttt{;}\ \textit{lit}\texttt{)} \\
\textit{quant} & ::= & \texttt{exists} \mid \texttt{forall} \\
\textit{var} & ::= & \text{(A string of ASCII letters, digits, and underscores)} \\
\textit{gvar} & ::= & \text{(A string of ASCII letters, digits, and underscores)} \\
\textit{nl} & ::= & \text{newline} \\
\textit{lit} & ::= & \textit{var} \mid \texttt{-}\textit{var} \mid \textit{gvar} \mid \texttt{-}\textit{gvar} \\
\textit{ngate\_type} & ::= & \texttt{and} \mid \texttt{or}
\end{array}
$$

2.2 Details

Comments. For the sake of readability comments are not included in the BNF description above. Any line that begins with the "`#`" character is a comment line and may be ignored.

Output. Gate definitions not connected to the output shall be ignored. The output must be a gate variable.

Separators. A QCIRfile may contain arbitrary many whitespaces, newlines and tabs. They only serve for formatting the formula or seperating tokens and may be ignored.

Gate Variables. Gate variables must be defined before they are used in the definition of another gate. For example, if the gate definitions include "`g1 = and(v1, v2)`" and "`g2 = or(g1, v3)`", then the definition of `g1` must come before the definition of `g2`. Note that this requirement ensures that the circuit graph is acyclic. Redefinition of gate variables is not allowed, i.e., a gate variable may not occur in `free` or as quantified variable and it occurs exactly once on the left-hand side of `=`.

Truth Constants. An `and` gate with zero inputs represents the constant `true`. An `or` gate with zero inputs represents the constant `false`.

Quantifier Alternations. Unlike QDIMACS, consecutive quantifier blocks of the same type are allowed. If non-gate variables are not explicitly quantified or included in the free variables block, they are assumed to be free variables.

Case-Sensitivity. Keywords (e.g., "`and`") are case-insensitive. Variable names are case-sensitive.

If-then-else (ite). `ite`(x, y, z) is an *if-then-else* gate; it is logically equivalent to $(x \wedge y) \vee (\neg x \wedge z)$.

Format ID. If the format id is followed by a number n, this indicates that the formula contains at most n different variables. If a variable is quantified m times, it is counted m times for establishing n. If n is given, the formula is assumed to be in cleansed form (see below).

Prenex Form. The format does not distinguish between formulas in prenex form and non-prenex form. Tools which process formulas in prenex form only, should use a prenexing tool in order to obtain the intended structure. A formula in prenex form can be written down as follows (no quantifier gates are allowed) where *gate_exp* are *and*, *or*, *ite*, or *xor* expressions.

```
#QCIR-G14
quant(var, ..., var)
⋮
quant(var, ..., var)
output(lit)
var = gate_exp
⋮
var = gate_exp
```

Free Variables. Tools that do not support free variables may assume that these variables are existentially quantified in the outermost quantifier block.

2.3 Cleansed Form

The cleansed form of a QCIR formula is a formula obeying the following syntactical restrictions.

- A variable is only quantified once.
- A variable is either quantified or free.
- All free variables are declared in the free block.
- Quantifiers and logical operators are only spelled in lower case letters.
- Gate-variables which have a subformula with quantification are used at most once.
- The format-id is followed by the number of variables n occurring in the formula (input + gate variables). The names of variables are integers smaller or equal to n.
- A cleansed formula does not contain `xor` gates.
- A cleansed formula does not contain `ite` gates.

A tool for transforming an arbitrary QCIRformula to its cleansed form will be provided at the Gallery Webpage. Note that cleansed formulas might blow up in size.

3 Examples

3.1 Formula in Prenex Form

A formula in prenex form looks as follows:

```
#QCIR-G14
forall(v1)
exists(v2, v3)
output(g3)
g1 = and(v1, v2)
g2 = and(-v1, -v2, v3)
g3 = or(g1, g2)
```

$$\forall v_1.\exists v_2.\exists v_3.\ \underbrace{\underbrace{(v_1 \wedge v_2)}_{g_1} \vee \underbrace{(\neg v_1 \wedge \neg v_2 \wedge v_3)}_{g_2}}_{g_3}$$

As seen above, a file in QCIR format consists of four parts: (1) format identification, (2) a quantifier prefix, (3) identification of the circuit output, and (4) gate definitions. In general, a formula in QCIR format has the following form:

3.2 Formula in Non-Prenex Form

A formula in non-prenex form looks as follows:

```
#QCIR-G14

forall(z)

output(g3)

g1 = and(x1, x2, z)

g2 = exists(x1, x2; g1)

g3 = or(z, g2)
```

$$\forall z.\ \overbrace{z \vee \exists x_1.\exists x_2.\ \underbrace{(x_1 \wedge x_2 \wedge z)}_{g_1}}^{g_2}$$

3.3 Formula in Cleansed Form

The formula from the previous section has the following cleansed form:

```
#QCIR-G14 6

forall(3)

output(4)

5 = and(1, 2, 3)

6 = exists(1, 2; 5)

4 = or(3, 6)
```

4 Beyond this Standard

This is a collection of topics to be handled in later versions of this document.

- Format of proofs (e.g., resolution)
- Format of Skolem/Herbrand functions
- Output if formula has free variables
- More detailed return values

5 Disclaimer

This version of QCIRwas prepared in the context of QBF Gallery 2014 and uses many parts of an internal version of the QCIRformat provided by Alexandra Goultiaeva and Will Klieber. The version QCIRis published and maintained by the organizers of the QBF Gallery 2014. Please send any comments or feedback to `qbfgallery2014@easychair.org`.