Formal Semantics of Kconfig for "Finding Broken Linux Configuration Specifications by Statically Analyzing the Kconfig Language"

JEHO OH*, University of Texas at Austin, USA
NECIP FAZIL YILDIRAN*, University of Central Florida, USA
JULIAN BRAHA, University of Central Florida, USA
PAUL GAZZILLO, University of Central Florida, USA

Contents

1 SYNTAX OF KCONFIG

Figure 1 describes the formal syntax of Kconfig, which includes all the constructs of Kconfig except `source`, `assign`, and `implies` construct. `source` statement merely merges statements defined in different files. `assign` statements is part of the preprocessor that runs before the conversion into a model. `implies` is a new construct for Linux version 5.5 and was not applied to the version we used (version 5.4.4).

As mentioned in the paper, Kconfig specification is insensitive to the ordering of *type*, *constraints*, and *select* within a statement. Our syntax defines ordering on these to reduce the number of semantic rules needed.

kconfig ::= **mainmenu** *word statement*+ | *statement*+
statement ::= *config* | *choice* | *if* | *menu* | *menuconfig*
config ::= **config** *symbol* **bool** *constrnts select** | **config** *symbol* **bool** *constrnts select** **option module**
| **config** *symbol* **tristate** *constrnts select** | **config** *symbol* **tristate** *constrnts select** **option module**
| **config** *symbol* **int** *constrnts* | **config** *symbol* **hex** *constrnts* | **config** *symbol* **string** *constrnts*
menuconfig ::= **menuconfig** *symbol type constrnts select**
choice ::= **choice bool** *constrnts config*+ **endchoice** | **choice bool** *constrnts* **optional** *config*+ **endchoice**
| **choice tristate** *constrnts config*+ **endchoice** | **choice tristate** *constrnts* **optional** *config*+ **endchoice**
if ::= **if** *expr statement*+ **endif**
menu ::= **menu** *visible*+ *depends*+ *statement*+ **endif**
type ::= **bool** | **tristate** | **int** | **hex** | **string**
constrnts ::= *prompt depends*+ *default*+ *range*+
prompt ::= **prompt** *word* | **prompt** *word* **if** *expr*
default ::= **default** *val* | **default** *val* **if** *expr* | **def_bool** *val* | **def_bool** *val* **if** *expr* | **def_tristate** *val* | **def_tristate** *val* **if** *expr*
range ::= **range** val_{lower} val_{upper} **if** *expr*
depends ::= **depends on** *expr*
visible ::= **visible if** *expr*
select ::= **select** *symbol* | **select** *symbol* **if** *expr*
expr ::= *expr* **&&** *expr* | *expr* **||** *expr* | **!** *expr*
| *symbol* **=** *symbol* | *symbol* **!=** *symbol* | *symbol* **<** *symbol* | *symbol* **<=** *symbol* | *symbol* **>** *symbol* | *symbol* **>=** *symbol*
| *symbol*
val ::= **y** | **m** | **n** | *decimal* | *hexadecimal* | *string* | ⊥

Fig. 1. Formal syntax of Kconfig.

*Co-first authors.

Authors' addresses: Jeho Oh, University of Texas at Austin, Austin, TX, USA, jeho.oh@utexas.edu; Necip Fazıl Yıldıran, University of Central Florida, Orlando, FL, USA, yildiran@knights.ucf.edu; Julian Braha, University of Central Florida, Orlando, FL, USA, julianbraha@knights.ucf.edu; Paul Gazzillo, University of Central Florida, Orlando, FL, USA, paul.gazzillo@ucf.edu.

2 SEMANTICS OF TRISTATE, INT, HEX, AND STRING OPTIONS

Figure 2 through Figure 5 extend the of semantics of Kconfig in Figure 3 of the paper for the int, hex, and string options.

Kconfig does not always support type checking of its operations. For example, *select* is only possible between `bool` and `tristate` options and other types cannot have reverse dependencies. However, the expression (*sym1 < sym2*) can have any type of options for *sym1* and *sym2*. We do not model the type checking, but consider that the specification is well-typed.

$$
\begin{aligned}
\Sigma &: \mathit{Symbols} \rightarrow \{\bot\} \cup \{\mathtt{n}, \mathtt{m}, \mathtt{y}\} \cup \mathbb{Z} \cup \mathbb{H} \cup \mathbb{S} \quad \mathit{where}\ \mathbb{Z} = \mathit{Set\ of\ all\ integers},\ \mathbb{H} = \mathit{Set\ of\ all\ hexadecimal\ numbers},\ \mathbb{S} = \mathit{Set\ of\ all\ strings} \\
\mathrm{E} &: \mathit{Constraints} \rightarrow \{\mathtt{true}, \mathtt{false}\} \\
\mathrm{E}_{tri} &: \mathit{Constraints} \rightarrow (\Sigma \rightarrow \{\mathtt{n}, \mathtt{m}, \mathtt{y}\}) \\
\mathrm{E}_{int} &: \mathit{Constraints} \rightarrow (\Sigma \rightarrow \{\bot\} \cup \mathbb{Z}) \\
\mathrm{E}_{hex} &: \mathit{Constraints} \rightarrow (\Sigma \rightarrow \{\bot\} \cup \mathbb{H}) \\
\mathrm{E}_{str} &: \mathit{Constraints} \rightarrow (\Sigma \rightarrow \{\bot\} \cup \mathbb{S})
\end{aligned}
$$

Fig. 2. Extended types for input and the valuation functions.

$$
\mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{tristate}\ \mathit{constrnts}\ \mathit{select}^*]]\sigma \triangleq
\begin{cases}
\mathtt{valid} & \textbf{if } (\sigma(\mathit{sym}) = \mathtt{y}) \wedge (\mathrm{R}_{tri}[[\mathit{kconfig}]](\sigma, \mathit{sym}) = \mathtt{y}) \\
\mathtt{valid} & \textbf{if } (\sigma(\mathit{sym}) = \mathtt{m}) \wedge (\mathrm{R}_{tri}[[\mathit{kconfig}]](\sigma, \mathit{sym}) = \mathtt{m}) \\
\mathtt{valid} & \textbf{if } (\mathrm{E}_{tri}[[\mathit{depends+}]]\sigma = \mathtt{y}) \wedge (\mathrm{E}_{tri}[[\mathit{prompt}]]\sigma = \mathtt{y}) \wedge (\mathrm{R}_{tri}[[\mathit{kconfig}]](\sigma, \mathit{sym}) = \mathtt{n}) \\
\mathtt{valid} & \textbf{if } (\mathrm{E}_{tri}[[\mathit{depends+}]]\sigma \in \{\mathtt{m}, \mathtt{y}\}) \wedge (\mathrm{E}_{tri}[[\mathit{prompt}]]\sigma \in \{\mathtt{m}, \mathtt{y}\}) \\
& \quad \wedge (\mathrm{R}_{tri}[[\mathit{kconfig}]](\sigma, \mathit{sym}) = \mathtt{n}) \wedge (\sigma(\mathit{sym}) \in \{\mathtt{n}, \mathtt{m}\}) \\
\mathtt{valid} & \textbf{if } (\mathrm{E}_{tri}[[\mathit{depends+}]]\sigma = \mathtt{y}) \wedge (\mathrm{E}_{tri}[[\mathit{prompt}]]\sigma = \mathtt{n}) \\
& \quad \wedge (\sigma(\mathit{sym}) = \mathrm{E}_{tri}[[\mathit{default+}]]\sigma) \\
\mathtt{valid} & \textbf{if } (\mathrm{E}_{tri}[[\mathit{depends+}]]\sigma = \mathtt{m}) \wedge (\mathrm{E}_{tri}[[\mathit{prompt}]]\sigma = \mathtt{n}) \\
& \quad \wedge (\mathrm{E}_{tri}[[\mathit{default+}]]\sigma \in \{\mathtt{m}, \mathtt{y}\}) \wedge (\sigma(\mathit{sym}) = \mathtt{m}) \\
\mathtt{valid} & \textbf{if } (\mathrm{E}_{tri}[[\mathit{depends+}]]\sigma = \mathtt{m}) \wedge (\mathrm{E}_{tri}[[\mathit{prompt}]]\sigma = \mathtt{n}) \\
& \quad \wedge (\mathrm{E}_{tri}[[\mathit{default+}]]\sigma = \mathtt{n}) \wedge (\sigma(\mathit{sym}) = \mathtt{n}) \\
\mathtt{valid} & \textbf{if } (\mathrm{E}_{tri}[[\mathit{depends+}]]\sigma = \mathtt{n}) \wedge (\mathrm{R}_{tri}[[\mathit{kconfig}]](\sigma, \mathit{sym}) = \mathtt{n}) \wedge (\sigma(\mathit{sym}) = \mathtt{n}) \\
\mathtt{invalid} & \text{otherwise}
\end{cases}
$$

$$
\mathrm{S}[[\textbf{choice tristate}\ \mathit{constrnts}\ \mathit{config+}\ \textbf{endchoice}]]\sigma \triangleq
\begin{cases}
\mathtt{valid} & \textbf{if } (\textsc{Enabled}[[\mathit{config+}]]\sigma = 1) \wedge (\mathrm{S}[[\mathit{config+}]]\sigma = \mathtt{valid}) \\
& \quad \wedge (\mathrm{E}_{int}[[\mathit{depends+}]]\sigma = \mathtt{y}) \wedge (\mathrm{E}_{int}[[\mathit{prompt}]]\sigma = \mathtt{y}) \\
\mathtt{valid} & \textbf{if } (\textsc{Enabled}[[\mathit{config+}]]\sigma = 0) \wedge (\mathrm{S}[[\mathit{config+}]]\sigma = \mathtt{valid}) \\
& \quad \wedge (\mathrm{E}_{int}[[\mathit{depends+}]]\sigma = \mathtt{y}) \wedge (\mathrm{E}_{int}[[\mathit{prompt}]]\sigma = \mathtt{y}) \\
\mathtt{valid} & \textbf{if } (\textsc{Enabled}[[\mathit{config+}]]\sigma = 0) \wedge (\mathrm{E}_{tri}[[\mathit{depends+}]]\sigma \in \{\mathtt{n}, \mathtt{m}\}) \wedge (\mathrm{E}_{tri}[[\mathit{prompt}]]\sigma \in \{\mathtt{n}, \mathtt{m}\}) \\
\mathtt{valid} & \textbf{if } (\textsc{Enabled}[[\mathit{config+}]]\sigma = 0) \wedge \neg \bigvee_{\mathit{constrnts}_i \in \mathit{config+}} (\mathrm{E}_{tri}[[\mathit{constrnts}_i]]\sigma = \mathtt{y}) \\
\mathtt{invalid} & \text{otherwise}
\end{cases}
$$

$$
\mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{int}\ \mathit{constrnts}]]\sigma \triangleq
\begin{cases}
\mathtt{valid} & \textbf{if } \mathrm{E}[[\mathit{depends+}]]\sigma \wedge \mathrm{E}[[\mathit{prompt}]]\sigma \wedge \mathrm{E}[[\mathit{range+}]]\sigma \wedge (\sigma(\mathit{sym}) \neq \bot) \wedge (\sigma(\mathit{sym}) \in \mathbb{Z}) \\
\mathtt{valid} & \textbf{if } \mathrm{E}[[\mathit{depends+}]]\sigma \wedge \neg\mathrm{E}[[\mathit{prompt}]]\sigma \wedge \mathrm{E}[[\mathit{range+}]]\sigma \wedge (\sigma(\mathit{sym}) = \mathrm{E}_{int}[[\mathit{default+}]]\sigma) \\
\mathtt{valid} & \textbf{if } \neg\mathrm{E}[[\mathit{depends+}]]\sigma \wedge (\sigma(\mathit{sym}) = \bot) \\
\mathtt{invalid} & \text{otherwise}
\end{cases}
$$

$$
\mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{hex}\ \mathit{constrnts}]]\sigma \triangleq
\begin{cases}
\mathtt{valid} & \textbf{if } \mathrm{E}[[\mathit{depends+}]]\sigma \wedge \mathrm{E}[[\mathit{prompt}]]\sigma \wedge \mathrm{E}[[\mathit{range+}]]\sigma \wedge (\sigma(\mathit{sym}) \neq \bot) \wedge (\sigma(\mathit{sym}) \in \mathbb{H}) \\
\mathtt{valid} & \textbf{if } \mathrm{E}[[\mathit{depends+}]]\sigma \wedge \neg\mathrm{E}[[\mathit{prompt}]]\sigma \wedge \mathrm{E}[[\mathit{range+}]]\sigma \wedge (\sigma(\mathit{sym}) = \mathrm{E}_{hex}[[\mathit{default+}]]\sigma) \\
\mathtt{valid} & \textbf{if } \neg\mathrm{E}[[\mathit{depends+}]]\sigma \wedge (\sigma(\mathit{sym}) = \bot) \\
\mathtt{invalid} & \text{otherwise}
\end{cases}
$$

$$
\mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{string}\ \mathit{constrnts}]]\sigma \triangleq
\begin{cases}
\mathtt{valid} & \textbf{if } \mathrm{E}[[\mathit{depends+}]]\sigma \wedge \mathrm{E}[[\mathit{prompt}]]\sigma \wedge (\sigma(\mathit{sym}) \neq \bot) \wedge (\sigma(\mathit{sym}) \in \mathbb{S}) \\
\mathtt{valid} & \textbf{if } \mathrm{E}[[\mathit{depends+}]]\sigma \wedge \neg\mathrm{E}[[\mathit{prompt}]]\sigma \wedge (\sigma(\mathit{sym}) = \mathrm{E}_{str}[[\mathit{default+}]]\sigma) \\
\mathtt{valid} & \textbf{if } \neg\mathrm{E}[[\mathit{depends+}]]\sigma \wedge (\sigma(\mathit{sym}) = \bot) \\
\mathtt{invalid} & \text{otherwise}
\end{cases}
$$

Fig. 3. Extended direct dependency rules for tristate, int, hex, and string options.

$$\mathrm{R}_{tri}[\![\textit{kconfig}]\!](\sigma, s) \triangleq \begin{cases} \mathtt{y} & \textbf{if } (\mathrm{R}_{tri}[\![\textit{statement}]\!](\sigma, s) = \mathtt{y}) \text{ for any } \textit{statement} \in \textit{kconfig} \\ \mathtt{n} & \textbf{if } (\mathrm{R}_{tri}[\![\textit{statement}]\!](\sigma, s) = \mathtt{n}) \text{ for all } \textit{statement} \in \textit{kconfig} \\ \mathtt{m} & \text{otherwise} \end{cases}$$

$$\mathrm{R}_{tri}[\![\texttt{config}\ \textit{sym}\ \texttt{tristate}\ \textit{constrnts select}^*]\!](\sigma, s) \triangleq \begin{cases} \mathtt{y} & \textbf{if } (\sigma(\textit{sym}) = \mathtt{y}) \land (\mathrm{E}_{tri}[\![\textit{depends}+]\!]\sigma = \mathtt{y}) \land (\mathrm{R}_{tri}[\![\textit{select}^*]\!](\sigma, s) = \mathtt{y}) \\ \mathtt{m} & \textbf{if } (\sigma(\textit{sym}) = \mathtt{m}) \land (\mathrm{E}_{tri}[\![\textit{depends}+]\!]\sigma \in \{\mathtt{m}, \mathtt{y}\}) \land (\mathrm{R}_{tri}[\![\textit{select}^*]\!](\sigma, s) \in \{\mathtt{m}, \mathtt{y}\}) \\ \mathtt{n} & \text{otherwise} \end{cases}$$

$$\mathrm{R}_{tri}[\![\texttt{select}\ \textit{sym}\ \texttt{if}\ \textit{expr select}^*]\!](\sigma, s) \triangleq \begin{cases} \mathtt{y} & \textbf{if } \mathrm{E}_{tri}[\![\textit{expr}]\!]\sigma = \mathtt{y} \land (\textit{sym} = s) \\ \mathtt{m} & \textbf{if } \mathrm{E}_{tri}[\![\textit{expr}]\!]\sigma = \mathtt{m} \land (\textit{sym} = s) \\ \mathrm{R}_{tri}[\![\textit{select}^*]\!](\sigma, s) & \textbf{if } \textit{default}^* \neq \emptyset \\ \mathtt{n} & \text{otherwise} \end{cases}$$

$$\mathrm{R}_{tri}[\![\texttt{choice}\ \texttt{tristate}\ \textit{constrnts config}+\ \texttt{endchoice}]\!](\sigma, s) \triangleq \begin{cases} \mathtt{y} & \textbf{if } (\mathrm{R}_{tri}[\![\textit{statement}]\!](\sigma, s) = \mathtt{y}) \text{ for any } \textit{statement} \in \textit{config}+ \\ \mathtt{n} & \textbf{if } (\mathrm{R}_{tri}[\![\textit{statement}]\!](\sigma, s) = \mathtt{n}) \text{ for all } \textit{statement} \in \textit{config}+ \\ \mathtt{m} & \text{otherwise} \end{cases}$$

Fig. 4. Extended reverse dependency rules for tristate, int, hex, and string options.

$$E_{tri}\llbracket \textbf{prompt}\ \mathit{word}\ \textbf{if}\ \mathit{expr} \rrbracket\sigma \triangleq E_{tri}\llbracket \mathit{expr} \rrbracket\sigma$$

$$E_{tri}\llbracket \mathit{depends+} \rrbracket\sigma \triangleq \begin{cases} \mathtt{y} & \textbf{if } (E_{tri}\llbracket \mathit{expr} \rrbracket\sigma = \mathtt{y}) \text{ for all } \mathit{expr} \in \mathit{depends+} \\ \mathtt{n} & \textbf{if } (E_{tri}\llbracket \mathit{expr} \rrbracket\sigma = \mathtt{n}) \text{ for any } \mathit{expr} \in \mathit{depends+} \\ \mathtt{m} & \text{otherwise} \end{cases}$$

$$E_{tri}\llbracket \textbf{default}\ \mathit{val}\ \textbf{if}\ \mathit{expr}\ \mathit{default}^* \rrbracket\sigma \triangleq \begin{cases} \mathit{val} & \textbf{if } (E_{tri}\llbracket \mathit{expr} \rrbracket\sigma = \mathtt{y}) \land (\mathit{val} \in \{\mathtt{n}, \mathtt{m}, \mathtt{y}\}) \\ \mathtt{m} & \textbf{if } (E_{tri}\llbracket \mathit{expr} \rrbracket\sigma = \mathtt{m}) \land (\mathit{val} \in \{\mathtt{m}, \mathtt{y}\}) \\ \mathtt{n} & \textbf{if } (E_{tri}\llbracket \mathit{expr} \rrbracket\sigma = \mathtt{m}) \land (\mathit{val} = \mathtt{n}) \\ E_{tri}\llbracket \mathit{default}^* \rrbracket\sigma & \textbf{if } \mathit{default}^* \neq \emptyset \\ \mathtt{n} & \text{otherwise} \end{cases}$$

$$E_{int}\llbracket \textbf{default}\ \mathit{val}\ \textbf{if}\ \mathit{expr}\ \mathit{default}^* \rrbracket\sigma \triangleq \begin{cases} \mathit{val} & \textbf{if } E\llbracket \mathit{expr} \rrbracket\sigma \land (\mathit{val} \in \mathbb{Z}) \\ E\llbracket \mathit{default}^* \rrbracket\sigma & \textbf{if } \mathit{default}^* \neq \emptyset \\ \bot & \text{otherwise} \end{cases}$$

$$E_{hex}\llbracket \textbf{default}\ \mathit{val}\ \textbf{if}\ \mathit{expr}\ \mathit{default}^* \rrbracket\sigma \triangleq \begin{cases} \mathit{val} & \textbf{if } E\llbracket \mathit{expr} \rrbracket\sigma \land (\mathit{val} \in \mathbb{H}) \\ E\llbracket \mathit{default}^* \rrbracket\sigma & \textbf{if } \mathit{default}^* \neq \emptyset \\ \bot & \text{otherwise} \end{cases}$$

$$E_{str}\llbracket \textbf{default}\ \mathit{val}\ \textbf{if}\ \mathit{expr}\ \mathit{default}^* \rrbracket\sigma \triangleq \begin{cases} \mathit{val} & \textbf{if } E\llbracket \mathit{expr} \rrbracket\sigma \land (\mathit{val} \in \mathbb{S}) \\ E\llbracket \mathit{default}^* \rrbracket\sigma & \textbf{if } \mathit{default}^* \neq \emptyset \\ \bot & \text{otherwise} \end{cases}$$

$$E\llbracket \textbf{range}\ a\ b\ \textbf{if}\ \mathit{expr}\ \mathit{range}^* \rrbracket\sigma \triangleq \begin{cases} a \leq \sigma(\mathit{sym}) \leq b & \textbf{if } E\llbracket \mathit{expr} \rrbracket\sigma \land ((\sigma(\mathit{sym}) \in \mathbb{Z}) \lor (\sigma(\mathit{sym}) \in \mathbb{H})) \\ E\llbracket \mathit{range}^* \rrbracket\sigma & \textbf{if } \mathit{range}^* \neq \emptyset \\ \mathtt{true} & \text{otherwise} \end{cases}$$

$$E_{tri}\llbracket \mathit{expr}_1\ \textbf{\&\&}\ \mathit{expr}_2 \rrbracket\sigma \triangleq \begin{cases} \mathtt{y} & \textbf{if } (E_{tri}\llbracket \mathit{expr}_1 \rrbracket\sigma = \mathtt{y}) \land (E_{tri}\llbracket \mathit{expr}_2 \rrbracket\sigma = \mathtt{y}) \\ \mathtt{n} & \textbf{if } (E_{tri}\llbracket \mathit{expr}_1 \rrbracket\sigma = \mathtt{n}) \lor (E_{tri}\llbracket \mathit{expr}_2 \rrbracket\sigma = \mathtt{n}) \\ \mathtt{m} & \text{otherwise} \end{cases}$$

$$E_{tri}\llbracket \mathit{expr}_1\ \textbf{||}\ \mathit{expr}_2 \rrbracket\sigma \triangleq \begin{cases} \mathtt{y} & \textbf{if } (E_{tri}\llbracket \mathit{expr}_1 \rrbracket\sigma = \mathtt{y}) \lor (E_{tri}\llbracket \mathit{expr}_2 \rrbracket\sigma = \mathtt{y}) \\ \mathtt{n} & \textbf{if } (E_{tri}\llbracket \mathit{expr}_1 \rrbracket\sigma = \mathtt{n}) \land (E_{tri}\llbracket \mathit{expr}_2 \rrbracket\sigma = \mathtt{n}) \\ \mathtt{m} & \text{otherwise} \end{cases}$$

$$E_{tri}\llbracket \textbf{!}\ \mathit{expr} \rrbracket\sigma \triangleq \begin{cases} \mathtt{y} & \textbf{if } E_{tri}\llbracket \mathit{expr} \rrbracket\sigma = \mathtt{n} \\ \mathtt{n} & \textbf{if } E_{tri}\llbracket \mathit{expr} \rrbracket\sigma = \mathtt{y} \\ \mathtt{n} & \text{otherwise} \end{cases}$$

$$E_{tri}\llbracket \mathit{sym} \rrbracket\sigma \triangleq \begin{cases} \sigma(\mathit{sym}) & \textbf{if } \sigma(\mathit{sym}) \in \{\mathtt{n}, \mathtt{m}, \mathtt{y}\} \\ \mathtt{n} & \text{otherwise} \end{cases}$$

$$E\llbracket \mathit{sym1}\ \textbf{=}\ \mathit{sym2} \rrbracket\sigma \triangleq \sigma(\mathit{sym}1) = \sigma(\mathit{sym}2)$$

$$E\llbracket \mathit{sym1}\ \textbf{!=}\ \mathit{sym2} \rrbracket\sigma \triangleq \sigma(\mathit{sym}1) \neq \sigma(\mathit{sym}2)$$

$$E\llbracket \mathit{sym1}\ \textbf{<}\ \mathit{sym2} \rrbracket\sigma \triangleq \sigma(\mathit{sym}1) < \sigma(\mathit{sym}2)$$

$$E\llbracket \mathit{sym1}\ \textbf{<=}\ \mathit{sym2} \rrbracket\sigma \triangleq \sigma(\mathit{sym}1) \leq \sigma(\mathit{sym}2)$$

$$E\llbracket \mathit{sym1}\ \textbf{>}\ \mathit{sym2} \rrbracket\sigma \triangleq \sigma(\mathit{sym}1) > \sigma(\mathit{sym}2)$$

$$E\llbracket \mathit{sym1}\ \textbf{>=}\ \mathit{sym2} \rrbracket\sigma \triangleq \sigma(\mathit{sym}1) \geq \sigma(\mathit{sym}2)$$

$$E\llbracket \mathit{sym} \rrbracket\sigma \triangleq \begin{cases} \mathtt{true} & \textbf{if } \sigma(\mathit{sym}) = \mathtt{y} \\ \mathtt{false} & \text{otherwise} \end{cases}$$

Fig. 5. Extended expression evaluation rules for tristate, int, hex, and string options.

3 SUPPLEMENTAL SEMANTIC RULES FOR KCONFIG

While the paper shows the core language for `bool` configuration options, Figure 6 describes additional core rules for choice with the **`optional`** keyword, which was not described in the paper for brevity.

Figure 7 through Figure 16 describe the rules for syntactic sugar. The rules in the paper and this supplemental material comprises all the rules for the `bool` configuration options.

$$\mathrm{S}[[\textbf{choice bool}\ \mathit{constrnts}\ \mathit{config}\texttt{+}\ \textbf{optional endchoice}]]\sigma \triangleq \begin{cases} \texttt{valid} & \textbf{if } (\textsc{Enabled}[[\mathit{config}\texttt{+}]]\sigma = 1) \land (\mathrm{S}[[\mathit{config}\texttt{+}]]\sigma = \texttt{valid}) \\ & \quad \land \mathrm{E}[[\mathit{depends}\texttt{+}]]\sigma \land \mathrm{E}[[\mathit{prompt}]]\sigma \\ \texttt{valid} & \textbf{if } (\textsc{Enabled}[[\mathit{config}\texttt{+}]]\sigma = 0) \\ \texttt{invalid} & \text{otherwise} \end{cases}$$

$$\mathrm{R}[[\textbf{choice bool}\ \mathit{constrnts}\ \mathit{config}\texttt{+}\ \textbf{optional endchoice}]](\sigma, s) \triangleq \mathrm{R}[[\textbf{choice bool}\ \mathit{constrnts}\ \mathit{config}\texttt{+}\ \textbf{endchoice}]](\sigma, s)$$

Fig. 6. Rules for `choice` statement with `optional` keyword.

$$\mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{bool}\ \mathit{depends}\texttt{+}\ \mathit{default}\texttt{+}\ \mathit{select}^*]]\sigma \triangleq \mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{bool prompt}\ \mathit{word}\ \textbf{if}\ \texttt{false}\ \mathit{depends}\texttt{+}\ \mathit{default}\texttt{+}\ \mathit{select}^*]]\sigma$$

$$\mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{bool}\ \mathit{prompt}\ \mathit{default}\texttt{+}\ \mathit{select}^*]]\sigma \triangleq \mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{bool}\ \mathit{prompt}\ \textbf{depends on}\ \texttt{true}\ \mathit{default}\texttt{+}\ \mathit{select}^*]]\sigma$$

$$\mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{bool}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \mathit{select}^*]]\sigma \triangleq \mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{bool}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \textbf{default}\ \texttt{n}\ \textbf{if}\ \texttt{true}\ \mathit{select}^*]]\sigma$$

$$\mathrm{R}[[\textbf{config}\ \mathit{sym}\ \textbf{bool}\ \mathit{depends}\texttt{+}\ \mathit{default}\texttt{+}\ \mathit{select}^*]](\sigma, s) \triangleq \mathrm{R}[[\textbf{config}\ \mathit{sym}\ \textbf{bool prompt}\ \mathit{word}\ \textbf{if}\ \texttt{false}\ \mathit{depends}\texttt{+}\ \mathit{default}\texttt{+}\ \mathit{select}^*]](\sigma, s)$$

$$\mathrm{R}[[\textbf{config}\ \mathit{sym}\ \textbf{bool}\ \mathit{prompt}\ \mathit{default}\texttt{+}\ \mathit{select}^*]](\sigma, s) \triangleq \mathrm{R}[[\textbf{config}\ \mathit{sym}\ \textbf{bool}\ \mathit{prompt}\ \textbf{depends on}\ \texttt{true}\ \mathit{default}\texttt{+}\ \mathit{select}^*]](\sigma, s)$$

$$\mathrm{R}[[\textbf{config}\ \mathit{sym}\ \textbf{bool}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \mathit{select}^*]](\sigma, s) \triangleq \mathrm{R}[[\textbf{config}\ \mathit{sym}\ \textbf{bool}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \textbf{default}\ \texttt{n}\ \textbf{if}\ \texttt{true}\ \mathit{select}^*]](\sigma, s)$$

$$\mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{tristate}\ \mathit{depends}\texttt{+}\ \mathit{default}\texttt{+}\ \mathit{select}^*]]\sigma \triangleq \mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{tristate prompt}\ \mathit{word}\ \textbf{if}\ \texttt{false}\ \mathit{depends}\texttt{+}\ \mathit{default}\texttt{+}\ \mathit{select}^*]]\sigma$$

$$\mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{tristate}\ \mathit{prompt}\ \mathit{default}\texttt{+}\ \mathit{select}^*]]\sigma \triangleq \mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{tristate}\ \mathit{prompt}\ \textbf{depends on}\ \texttt{true}\ \mathit{default}\texttt{+}\ \mathit{select}^*]]\sigma$$

$$\mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{tristate}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \mathit{select}^*]]\sigma \triangleq \mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{tristate}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \textbf{default}\ \texttt{n}\ \textbf{if}\ \texttt{true}\ \mathit{select}^*]]\sigma$$

$$\mathrm{R}[[\textbf{config}\ \mathit{sym}\ \textbf{tristate}\ \mathit{depends}\texttt{+}\ \mathit{default}\texttt{+}\ \mathit{select}^*]](\sigma, s) \triangleq \mathrm{R}[[\textbf{config}\ \mathit{sym}\ \textbf{tristate prompt}\ \mathit{word}\ \textbf{if}\ \texttt{false}\ \mathit{depends}\texttt{+}\ \mathit{default}\texttt{+}\ \mathit{select}^*]](\sigma, s)$$

$$\mathrm{R}[[\textbf{config}\ \mathit{sym}\ \textbf{tristate}\ \mathit{prompt}\ \mathit{default}\texttt{+}\ \mathit{select}^*]](\sigma, s) \triangleq \mathrm{R}[[\textbf{config}\ \mathit{sym}\ \textbf{tristate}\ \mathit{prompt}\ \textbf{depends on}\ \texttt{true}\ \mathit{default}\texttt{+}\ \mathit{select}^*]](\sigma, s)$$

$$\mathrm{R}[[\textbf{config}\ \mathit{sym}\ \textbf{tristate}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \mathit{select}^*]](\sigma, s) \triangleq \mathrm{R}[[\textbf{config}\ \mathit{sym}\ \textbf{tristate}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \textbf{default}\ \texttt{n}\ \textbf{if}\ \texttt{true}\ \mathit{select}^*]](\sigma, s)$$

$$\mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{int}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \mathit{range}\texttt{+}]]\sigma \triangleq \mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{int}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \textbf{default}\ \bot\ \textbf{if}\ \texttt{true}\ \mathit{range}\texttt{+}]]\sigma$$

$$\mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{hex}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \mathit{range}\texttt{+}]]\sigma \triangleq \mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{hex}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \textbf{default}\ \bot\ \textbf{if}\ \texttt{true}\ \mathit{range}\texttt{+}]]\sigma$$

$$\mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{string}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \mathit{range}\texttt{+}]]\sigma \triangleq \mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{string}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \textbf{default}\ \bot\ \textbf{if}\ \texttt{true}\ \mathit{range}\texttt{+}]]\sigma$$

$$\mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{int}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \mathit{default}\texttt{+}]]\sigma \triangleq \mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{int}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \mathit{default}\ \texttt{+}\ \textbf{range}\ \text{INT\_MIN INT\_MAX}\ \textbf{if}\ \texttt{true}]]\sigma$$

$$\mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{hex}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \mathit{default}\texttt{+}]]\sigma \triangleq \mathrm{S}[[\textbf{config}\ \mathit{sym}\ \textbf{hex}\ \mathit{prompt}\ \mathit{depends}\texttt{+}\ \mathit{default}\ \texttt{+}\ \textbf{range}\ \text{HEX\_MIN HEX\_MAX}\ \textbf{if}\ \texttt{true}]]\sigma$$

Fig. 7. Example of rules for statements with omitted constraints.

$$\mathrm{S}[\![\textbf{choice}\ \mathit{constrnts}\ \textbf{config}\ \mathit{sym}\ \textbf{bool}\ \mathit{constrnts}\ \mathit{select}^*\ \mathit{config}^*\ \textbf{endchoice}]\!]\sigma \overset{\Delta}{=} \mathrm{S}[\![\textbf{choice}\ \textbf{bool}\ \mathit{constrnts}\ \textbf{config}\ \mathit{sym}\ \textbf{bool}\ \mathit{constrnts}\ \mathit{select}^*\ \mathit{config}^*\ \textbf{endchoice}]\!]\sigma$$

$$\mathrm{S}[\![\textbf{choice}\ \mathit{constrnts}\ \textbf{config}\ \mathit{sym}\ \textbf{tristate}\ \mathit{constrnts}\ \mathit{select}^*\ \mathit{config}^*\ \textbf{endchoice}]\!]\sigma \overset{\Delta}{=} \mathrm{S}[\![\textbf{choice}\ \textbf{tristate}\ \mathit{constrnts}\ \textbf{config}\ \mathit{sym}\ \textbf{tristate}\ \mathit{constrnts}\ \mathit{select}^*\ \mathit{config}^*\ \textbf{endchoice}]\!]\sigma$$

$$\mathrm{R}[\![\textbf{choice}\ \mathit{constrnts}\ \textbf{config}\ \mathit{sym}\ \textbf{bool}\ \mathit{constrnts}\ \mathit{select}^*\ \mathit{config}^*\ \textbf{endchoice}]\!](\sigma, s) \overset{\Delta}{=} \mathrm{R}[\![\textbf{choice}\ \textbf{bool}\ \mathit{constrnts}\ \textbf{config}\ \mathit{sym}\ \textbf{bool}\ \mathit{constrnts}\ \mathit{select}^*\ \mathit{config}^*\ \textbf{endchoice}]\!](\sigma, s)$$

$$\mathrm{R}[\![\textbf{choice}\ \mathit{constrnts}\ \textbf{config}\ \mathit{sym}\ \textbf{tristate}\ \mathit{constrnts}\ \mathit{select}^*\ \mathit{config}^*\ \textbf{endchoice}]\!](\sigma, s) \overset{\Delta}{=} \mathrm{R}[\![\textbf{choice}\ \textbf{tristate}\ \mathit{constrnts}\ \textbf{config}\ \mathit{sym}\ \textbf{tristate}\ \mathit{constrnts}\ \mathit{select}^*\ \mathit{config}^*\ \textbf{endchoice}]\!](\sigma, s)$$

Fig. 8. Rules for choice statements without type.

$$\mathrm{S}[\![\textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \mathit{config}\ \mathit{statement}{+}\ \textbf{endmenu}]\!]\sigma \overset{\Delta}{=} \mathrm{S}[\![\textbf{config}\ \mathit{sym}\ \mathit{type}\ \textbf{prompt}\ \mathit{word}_{config}\ \textbf{if}\ \mathit{expr}\ \textbf{\&\&}\ \textsc{Conjoin}[\![\mathit{visible}{+}]\!]\ \mathit{depends}{+}\ \mathit{depends}_{config}{+}\ \mathit{default}{+}\ \mathit{select}^*\ \textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \mathit{statement}{+}\ \textbf{endmenu}]\!]\sigma$$

$$\mathrm{S}[\![\textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \mathit{choice}\ \mathit{statement}{+}\ \textbf{endmenu}]\!]\sigma \overset{\Delta}{=} \mathrm{S}[\![\textbf{choice}\ \mathit{type}\ \textbf{prompt}\ \mathit{word}_{choice}\ \textbf{if}\ \mathit{expr}\ \textbf{\&\&}\ \textsc{Conjoin}[\![\mathit{visible}{+}]\!]\ \mathit{depends}{+}\ \mathit{depends}_{choice}{+}\ \mathit{default}{+}\ \mathit{config}{+}\ \textbf{endchoice}\ \textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \mathit{statement}{+}\ \textbf{endmenu}]\!]\sigma$$

$$\mathrm{S}[\![\textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \mathit{if}\ \mathit{statement}{+}\ \textbf{endmenu}]\!]\sigma \overset{\Delta}{=} \mathrm{S}[\![\textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \textbf{depends on}\ \mathit{expr}_{if}\ \mathit{statement}_{if}{+}\ \textbf{endmenu}\ \textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \mathit{statement}{+}\ \textbf{endmenu}]\!]\sigma$$

$$\mathrm{S}[\![\textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \mathit{menu}\ \mathit{statement}{+}\ \textbf{endmenu}]\!]\sigma \overset{\Delta}{=} \mathrm{S}[\![\textbf{menu}\ \mathit{word}_{menu}\ \mathit{visible}_{menu}\ \mathit{visible}{+}\ \mathit{depends}_{menu}{+}\ \mathit{depends}{+}\ \mathit{statement}_{menu}{+}\ \textbf{endmenu}\ \textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \mathit{statement}{+}\ \textbf{endmenu}]\!]\sigma$$

$$\mathrm{R}[\![\textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \mathit{config}\ \mathit{statement}{+}\ \textbf{endmenu}]\!](\sigma, s) \overset{\Delta}{=} \mathrm{R}[\![\textbf{config}\ \mathit{sym}\ \mathit{type}\ \textbf{prompt}\ \mathit{word}_{config}\ \textbf{if}\ \mathit{expr}\ \textbf{\&\&}\ \textsc{Conjoin}[\![\mathit{visible}{+}]\!]\ \mathit{depends}{+}\ \mathit{depends}_{config}{+}\ \mathit{default}{+}\ \mathit{select}^*\ \textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \mathit{statement}{+}\ \textbf{endmenu}]\!](\sigma, s)$$

$$\mathrm{R}[\![\textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \mathit{choice}\ \mathit{statement}{+}\ \textbf{endmenu}]\!](\sigma, s) \overset{\Delta}{=} \mathrm{R}[\![\textbf{choice}\ \mathit{type}\ \textbf{prompt}\ \mathit{word}_{choice}\ \textbf{if}\ \mathit{expr}\ \textbf{\&\&}\ \textsc{Conjoin}[\![\mathit{visible}{+}]\!]\ \mathit{depends}{+}\ \mathit{depends}_{choice}{+}\ \mathit{default}{+}\ \mathit{config}{+}\ \textbf{endchoice}\ \textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \mathit{statement}{+}\ \textbf{endmenu}]\!](\sigma, s)$$

$$\mathrm{R}[\![\textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \mathit{if}\ \mathit{statement}{+}\ \textbf{endmenu}]\!](\sigma, s) \overset{\Delta}{=} \mathrm{R}[\![\textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \textbf{depends on}\ \mathit{expr}_{if}\ \mathit{statement}_{if}{+}\ \textbf{endmenu}\ \textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \mathit{statement}{+}\ \textbf{endmenu}]\!](\sigma, s)$$

$$\mathrm{R}[\![\textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \mathit{menu}\ \mathit{statement}{+}\ \textbf{endmenu}]\!](\sigma, s) \overset{\Delta}{=} \mathrm{R}[\![\textbf{menu}\ \mathit{word}_{menu}\ \mathit{visible}_{menu}\ \mathit{visible}{+}\ \mathit{depends}_{menu}{+}\ \mathit{depends}{+}\ \mathit{statement}_{menu}{+}\ \textbf{endmenu}\ \textbf{menu}\ \mathit{word}\ \mathit{visible}{+}\ \mathit{depends}{+}\ \mathit{statement}{+}\ \textbf{endmenu}]\!](\sigma, s)$$

Fig. 9. Rules for menu statement.

$$\textsc{Conjoin}[\![\textbf{visible if}\ \mathit{expr}\ \mathit{visible}{+}]\!] \overset{\Delta}{=} \mathit{expr}\ \textbf{\&\&}\ \textsc{Conjoin}[\![\mathit{visible}{+}]\!]$$

Fig. 10. Generating a construct that conjoins all expr of visible constructs.

$$S[[\texttt{if}\ \mathit{expr\ config\ statement+}\ \texttt{endif}]]\sigma \triangleq S[[\texttt{config}\ \mathit{sym\ type\ constrnts}\ \texttt{depends on}\ \mathit{expr\ select^*}\ \texttt{if}\ \mathit{expr\ statement+}\ \texttt{endif}]]\sigma$$

$$S[[\texttt{if}\ \mathit{expr\ choice\ statement+}\ \texttt{endif}]]\sigma \triangleq S[[\texttt{choice}\ \mathit{type\ constrnts}\ \texttt{depends on}\ \mathit{expr\ config+}\ \texttt{endchoice}\ \texttt{if}\ \mathit{expr\ statement+}\ \texttt{endif}]]\sigma$$

$$S[[\texttt{if}\ \mathit{expr\ if\ statement+}\ \texttt{endif}]]\sigma \triangleq S[[\texttt{if}\ \mathit{expr}\ \texttt{\&\&}\ \mathit{expr_{if}\ statement_{if}+}\ \texttt{endif}\ \texttt{if}\ \mathit{expr\ statement+}\ \texttt{endif}]]\sigma$$

$$S[[\texttt{if}\ \mathit{expr\ menu\ statement+}\ \texttt{endif}]]\sigma \triangleq S[[\texttt{menu}\ \mathit{visible+\ depends+}\ \texttt{depends on}\ \mathit{expr\ statement_{menu}+}\ \texttt{endmenu}\ \texttt{if}\ \mathit{expr\ statement+}\ \texttt{endif}]]\sigma$$

$$R[[\texttt{if}\ \mathit{expr\ config\ statement+}\ \texttt{endif}]](\sigma,s) \triangleq R[[\texttt{config}\ \mathit{sym\ type\ constrnts}\ \texttt{depends on}\ \mathit{expr\ select^*}\ \texttt{if}\ \mathit{expr\ statement+}\ \texttt{endif}]](\sigma,s)$$

$$R[[\texttt{if}\ \mathit{expr\ choice\ statement+}\ \texttt{endif}]](\sigma,s) \triangleq R[[\texttt{choice}\ \mathit{type\ constrnts}\ \texttt{depends on}\ \mathit{expr\ config+}\ \texttt{endchoice}\ \texttt{if}\ \mathit{expr\ statement+}\ \texttt{endif}]](\sigma,s)$$

$$R[[\texttt{if}\ \mathit{expr\ if\ statement+}\ \texttt{endif}]](\sigma,s) \triangleq R[[\texttt{if}\ \mathit{expr}\ \texttt{\&\&}\ \mathit{expr_{if}\ statement_{if}+}\ \texttt{endif}\ \texttt{if}\ \mathit{expr\ statement+}\ \texttt{endif}]](\sigma,s)$$

$$R[[\texttt{if}\ \mathit{expr\ menu\ statement+}\ \texttt{endif}]](\sigma,s) \triangleq R[[\texttt{menu}\ \mathit{visible+\ depends+}\ \texttt{depends on}\ \mathit{expr\ statement_{menu}+}\ \texttt{endmenu}\ \texttt{if}\ \mathit{expr\ statement+}\ \texttt{endif}]](\sigma,s)$$

Fig. 11. Rules for if statement.

$$S[[\texttt{mainmenu}\ \mathit{word\ statement+}]]\sigma \triangleq S[[\mathit{statement+}]]\sigma$$

$$R[[\texttt{mainmenu}\ \mathit{word\ statement+}]](\sigma,s) \triangleq R[[\mathit{statement+}]](\sigma,s)$$

Fig. 12. Rules for mainmenu statement.

$$S[[\texttt{menuconfig}\ \mathit{sym\ type\ constrnts\ select^*}]]\sigma \triangleq S[[\texttt{config}\ \mathit{sym\ type\ constrnts\ select^*}]]\sigma$$

$$R[[\texttt{menuconfig}\ \mathit{sym\ type\ constrnts\ select^*}]](\sigma,s) \triangleq R[[\texttt{config}\ \mathit{sym\ type\ constrnts\ select^*}]](\sigma,s)$$

Fig. 13. Rules for menuconfig statement.

$$S[[\texttt{config}\ \mathit{sym}\ \texttt{def\_bool}\ \mathit{val}\ \texttt{if}\ \mathit{expr\ constrnts\ select^*}]]\sigma \triangleq S[[\texttt{config}\ \mathit{sym}\ \texttt{bool}\ \mathit{prompt\ depends+}\ \texttt{default}\ \mathrm{val}\ \texttt{if}\ \mathit{expr\ default+\ select^*}]]\sigma$$

$$R[[\texttt{config}\ \mathit{sym}\ \texttt{def\_bool}\ \mathit{val}\ \texttt{if}\ \mathit{expr\ constrnts\ select^*}]](\sigma,s) \triangleq R[[\texttt{config}\ \mathit{sym}\ \texttt{bool}\ \mathit{prompt\ depends+}\ \texttt{default}\ \mathrm{val}\ \texttt{if}\ \mathit{expr\ default+\ select^*}]](\sigma,s)$$

$$R[[\texttt{config}\ \mathit{sym}\ \texttt{def\_tristate}\ \mathit{val}\ \texttt{if}\ \mathit{expr\ constrnts\ select^*}]](\sigma,s) \triangleq R[[\texttt{config}\ \mathit{sym}\ \texttt{tristate}\ \mathit{prompt\ depends+}\ \texttt{default}\ \mathrm{val}\ \texttt{if}\ \mathit{expr\ default+\ select^*}]](\sigma,s)$$

$$S[[\texttt{config}\ \mathit{sym}\ \texttt{def\_tristate}\ \mathit{val}\ \texttt{if}\ \mathit{expr\ constrnts\ select^*}]]\sigma \triangleq S[[\texttt{config}\ \mathit{sym}\ \texttt{tristate}\ \mathit{prompt\ depends+}\ \texttt{default}\ \mathrm{val}\ \texttt{if}\ \mathit{expr\ default+\ select^*}]]\sigma$$

Fig. 14. Rules for config statement with def_bool or def_tristate construct.

$$S[[\texttt{config}\ \mathit{sym\ type\ word}\ \texttt{if}\ \mathit{expr\ constrnts\ select^*}]]\sigma \triangleq S[[\texttt{config}\ \mathit{sym\ type}\ \texttt{prompt}\ \mathit{word}\ \texttt{if}\ \mathit{expr\ constrnts\ select^*}]]\sigma$$

$$S[[\texttt{choice bool}\ \mathit{word}\ \texttt{if}\ \mathit{expr\ depends+\ default+\ config+}\ \texttt{end}]]\sigma \triangleq S[[\texttt{choice bool prompt}\ \mathit{word}\ \texttt{if}\ \mathit{expr\ depends+\ default+\ config+}\ \texttt{end}]]\sigma$$

$$S[[\texttt{choice tristate}\ \mathit{word}\ \texttt{if}\ \mathit{expr\ depends+\ default+\ config+}\ \texttt{end}]]\sigma \triangleq S[[\texttt{choice tristate prompt}\ \mathit{word}\ \texttt{if}\ \mathit{expr\ depends+\ default+\ config+}\ \texttt{end}]]\sigma$$

$$R[[\texttt{config}\ \mathit{sym}\ \texttt{bool}\ \mathit{word}\ \texttt{if}\ \mathit{expr\ constrnts\ select^*}]](\sigma,s) \triangleq R[[\texttt{config}\ \mathit{sym}\ \texttt{bool prompt}\ \mathit{word}\ \texttt{if}\ \mathit{expr\ constrnts\ select^*}]](\sigma,s)$$

$$R[[\texttt{config}\ \mathit{sym}\ \texttt{tristate}\ \mathit{word}\ \texttt{if}\ \mathit{expr\ constrnts\ select^*}]](\sigma,s) \triangleq R[[\texttt{config}\ \mathit{sym}\ \texttt{tristate prompt}\ \mathit{word}\ \texttt{if}\ \mathit{expr\ constrnts\ select^*}]](\sigma,s)$$

$$R[[\texttt{choice bool}\ \mathit{word}\ \texttt{if}\ \mathit{expr\ depends+\ default+\ config+}\ \texttt{end}]](\sigma,s) \triangleq R[[\texttt{choice bool prompt}\ \mathit{word}\ \texttt{if}\ \mathit{expr\ depends+\ default+\ config+}\ \texttt{end}]](\sigma,s)$$

$$R[[\texttt{choice tristate}\ \mathit{word}\ \texttt{if}\ \mathit{expr\ depends+\ default+\ config+}\ \texttt{end}]](\sigma,s) \triangleq R[[\texttt{choice tristate prompt}\ \mathit{word}\ \texttt{if}\ \mathit{expr\ depends+\ default+\ config+}\ \texttt{end}]](\sigma,s)$$

Fig. 15. Rules for config and choice statements that does not have prompt keyword.

$$\mathrm{E}[\![\textbf{prompt}\ \mathit{word}]\!]\sigma \triangleq \mathrm{E}[\![\textbf{prompt}\ \mathit{word}\ \textbf{if}\ \texttt{true}]\!]\sigma$$

$$\mathrm{E}[\![\textbf{default}\ \mathit{val}]\!]\sigma \triangleq \mathrm{E}[\![\textbf{default}\ \mathit{val}\ \textbf{if}\ \texttt{true}]\!]\sigma$$

$$\mathrm{E}_{int}[\![\textbf{default}\ \mathit{val}]\!]\sigma_{int} \triangleq \mathrm{E}_{int}[\![\textbf{default}\ \mathit{val}\ \textbf{if}\ \texttt{true}]\!]\sigma$$

$$\mathrm{E}_{hex}[\![\textbf{default}\ \mathit{val}]\!]\sigma_{hex} \triangleq \mathrm{E}_{hex}[\![\textbf{default}\ \mathit{val}\ \textbf{if}\ \texttt{true}]\!]\sigma$$

$$\mathrm{E}_{str}[\![\textbf{default}\ \mathit{val}]\!]\sigma_{str} \triangleq \mathrm{E}_{str}[\![\textbf{default}\ \mathit{val}\ \textbf{if}\ \texttt{true}]\!]\sigma$$

$$\mathrm{R}[\![\textbf{select}\ \mathit{sym}]\!](\sigma, s) \triangleq \mathrm{R}[\![\textbf{select}\ \mathit{sym}\ \textbf{if}\ \texttt{true}]\!](\sigma, s)$$

$$\mathrm{E}[\![\textbf{range}\ a\ b]\!]\sigma \triangleq \mathrm{E}[\![\textbf{range}\ a\ b\ \textbf{if}\ \texttt{true}]\!]\sigma$$

Fig. 16. Rules for `prompt`, `default`, `select` and, `range` without conditions.

4 SYMBOLIC VALUATION RULES

Figure 17 through Figure 21 show the valuation rules for the core language of `bool` configuration option.

$$\Gamma : \textit{Symbols} \rightarrow \textit{Symbolic values}$$
$$\Phi : \textit{Statements} \rightarrow (\Gamma \rightarrow formula)$$
$$\Phi_R : \textit{Statements} \rightarrow (\Gamma \times \textit{Symbols} \rightarrow formula)$$

Fig. 17. Types for conversion functions.

$$\Phi[[\textit{kconfig}]]\gamma \triangleq \bigwedge_{\textit{statement}_i \in \textit{kconfig}} \Phi[[\textit{statement}_i]]\gamma$$

$$\begin{aligned}
\Phi[[\textbf{config}\ \textit{sym}\ \textbf{bool}\ \textit{constrnts}\ \textit{select}^*]]\gamma \triangleq\ & (\Phi[[\textit{sym}]]\gamma \wedge \Phi_R[[\textit{kconfig}]](\gamma, \textit{sym})) \\
& \vee (\Phi[[\textit{depends+}]]\gamma \wedge \Phi[[\textit{prompt}]]\gamma \wedge \neg\Phi_R[[\textit{kconfig}]](\gamma, \textit{sym})) \\
& \vee ((\Phi[[\textit{sym}]]\gamma \wedge \Phi[[\textit{default+}]]\gamma \vee \neg\Phi[[\textit{sym}]]\gamma \wedge \neg\Phi[[\textit{default+}]]\gamma) \\
& \quad \wedge \Phi[[\textit{depends+}]]\gamma \wedge \neg\Phi[[\textit{prompt}]]\gamma \wedge \neg\Phi_R[[\textit{kconfig}]](\gamma, \textit{sym})) \\
& \vee (\neg\Phi[[\textit{sym}]]\gamma \wedge \neg\Phi[[\textit{depends+}]]\gamma \wedge \neg\Phi_R[[\textit{kconfig}]](\gamma, \textit{sym}))
\end{aligned}$$

$$\begin{aligned}
\Phi[[\textbf{choice}\ \textbf{bool}\ \textit{constrnts}\ \textit{config+}\ \textbf{endchoice}]] \triangleq\ & (\textsc{Onlyone}[[\textit{config+}]]\gamma \wedge \Phi[[\textit{config+}]]\gamma \wedge \Phi[[\textit{depends+}]]\gamma \wedge \Phi[[\textit{prompt}]]\gamma) \\
& \vee (\textsc{None}[[\textit{config+}]]\gamma \wedge \neg(\Phi[[\textit{depends+}]]\gamma \wedge \Phi[[\textit{prompt}]]\gamma)) \\
& \vee \left(\textsc{None}[[\textit{config+}]]\gamma \wedge \bigwedge_{\textit{constrnts}_i \in \textit{config+}} \neg\Phi[[\textit{constrnts}_i]]\gamma\right)
\end{aligned}$$

$$\begin{aligned}
\Phi[[\textbf{choice}\ \textbf{bool}\ \textit{constrnts}\ \textit{config+}\ \textbf{optional}\ \textbf{endchoice}]] \triangleq\ & (\textsc{Onlyone}[[\textit{config+}]]\gamma \wedge \Phi[[\textit{config+}]]\gamma \wedge \Phi[[\textit{depends+}]]\gamma \wedge \Phi[[\textit{prompt}]]\gamma) \\
& \vee (\textsc{None}[[\textit{config+}]]\gamma)
\end{aligned}$$

Fig. 18. Direct dependency conversion rules.

$$\Phi_R[[\textit{kconfig}]](\gamma, s) \triangleq \bigvee_{\textit{statement}_i \in \textit{kconfig}} \Phi_R[[\textit{statement}_i]](\gamma, s)$$

$$\Phi_R[[\textbf{config}\ \textit{sym}\ \textbf{bool}\ \textit{constrnts}\ \textit{select}^*]](\gamma, s) \triangleq \Phi_R[[\textit{select}^*]](\gamma, s) \wedge \Phi[[\textit{sym}]]\gamma \wedge \Phi[[\textit{depends+}]]\gamma$$

$$\Phi_R[[\textbf{select}\ \textit{sym}\ \textbf{if}\ \textit{expr}\ \textit{select}^*]](\gamma, s) \triangleq ((\textit{sym} = s) \wedge \Phi[[\textit{expr}]]) \vee \Phi_R[[\textit{select}^*]](\gamma, s)$$

$$\Phi_R[[\textbf{choice}\ \textbf{bool}\ \textit{constrnts}\ \textit{config+}\ \textbf{endchoice}]](\gamma, s) \triangleq \Phi_R[[\textit{config+}]](\gamma, s) \wedge \Phi[[\textit{depends+}]]\gamma \wedge \Phi[[\textit{prompt}]]\gamma$$

Fig. 19. Reverse dependency conversion rules.

$$\Phi[[\textbf{prompt}\ \mathit{word}\ \textbf{if}\ \mathit{expr}]]\gamma \stackrel{\Delta}{=} (\Phi[[\mathit{expr}]]\gamma)$$

$$\Phi[[\mathit{depends+}]]\gamma \stackrel{\Delta}{=} \bigwedge_{\mathit{expr}_i \in \mathit{depends+}} (\Phi[[\mathit{expr}_i]]\gamma)$$

$$\Phi[[\textbf{default}\ \mathit{val}\ \textbf{if}\ \mathit{expr}\ \mathit{default}^*]]\gamma \stackrel{\Delta}{=} (\mathit{val} \wedge \Phi[[\mathit{expr}]]\gamma) \vee (\Phi[[\mathit{default}^*]]\gamma \wedge \neg\Phi[[\mathit{expr}]]\gamma)$$

$$\Phi[[\mathit{expr}_1\ \textbf{\&\&}\ \mathit{expr}_2]]\gamma \stackrel{\Delta}{=} \Phi[[\mathit{expr}_1]]\gamma \wedge \Phi[[\mathit{expr}_2]]\gamma$$

$$\Phi[[\mathit{expr}_1\ \textbf{||}\ \mathit{expr}_2]]\gamma \stackrel{\Delta}{=} \Phi[[\mathit{expr}_1]]\gamma \vee \Phi[[\mathit{expr}_2]]\gamma$$

$$\Phi[[\ \textbf{!}\ \mathit{expr}]]\gamma \stackrel{\Delta}{=} \neg\Phi[[\mathit{expr}]]\gamma$$

$$\Phi[[\mathit{sym}]]\gamma \stackrel{\Delta}{=} \gamma(\mathit{sym})$$

Fig. 20. Evaluation conversion rules.

$$\textsc{OnlyOne}[[\mathit{config+}]]\gamma \stackrel{\Delta}{=} \bigwedge_{\substack{\mathit{sym}_i, \mathit{sym}_j \in \mathit{config+} \\ i \neq j}} (\neg\gamma(\mathit{sym}_i) \vee \neg\gamma(\mathit{sym}_j)) \wedge \bigvee_{\mathit{sym}_i \in \mathit{config+}} \gamma(\mathit{sym}_i)$$

$$\textsc{None}[[\mathit{config+}]]\gamma \stackrel{\Delta}{=} \bigwedge_{\mathit{sym}_i \in \mathit{config+}} \neg\gamma(\mathit{sym}_i)$$

Fig. 21. Counting enabled config options.