Redundancy Detection in Configuration Knowledge

Alexander Felfernig†, Florian Reinfrank†, Gerald Ninaus†, and Paul Blazek‡

†Graz University of Technology, Graz, Austria
‡cyLEDGE, Vienna, Austria
Contents

• Definitions
• Sequential
• CoreDiag
• Evaluation
Example Knowledge Base

- **Variables** ($V$) = \{type, fuel, skibag, 4-wheel, pdc\}
- **Domains** ($D$) = \{dom(type) = \{city, limo, combi, xdrive\},
  
  \begin{align*}
  \text{dom(fuel)} & = \{4l, 6l, 10l\}, \\
  \text{dom(skibag)} & = \{yes, no\}, \\
  \text{dom(4-wheel)} & = \{yes, no\}, \\
  \text{dom(pdc)} & = \{yes, no\}\}
  
- **Knowledge Base** ($C_{KB}$) = \{

  \begin{align*}
  c_1 : & \text{4-wheel = yes } \rightarrow \text{type = xdrive}, \\
  c_2 : & \text{skibag = yes } \rightarrow \text{type } \neq \text{city}, \\
  c_3 : & \text{fuel = 4l } \rightarrow \text{type = city}, \\
  c_4 : & \text{fuel = 6l } \rightarrow \text{type } \neq \text{xdrive}, \\
  c_5 : & \text{type = city } \rightarrow \text{fuel } \neq 101
  \end{align*}

- **Customer Requirements** ($C_R$) = \{

  \begin{align*}
  c_6 : & \text{4-wheel = no}, \\
  c_7 : & \text{fuel = 4l}, \\
  c_8 : & \text{type = city}, \\
  c_9 : & \text{skibag = no}, \\
  c_{10} : & \text{pdc = yes}
  \end{align*}

Redundancy Detection in Configuration Knowledge
Redundant Knowledge Base

\[ C'_{KB} = \{
\begin{align*}
  c_a: \text{skibag} \neq \text{no} & \rightarrow \text{type} = \text{limo} \lor \\
  & \text{type} = \text{combi} \lor \\
  & \text{type} = \text{xdrive}, \\
  c_1: \text{4-wheel} = \text{yes} & \rightarrow \text{type} = \text{xdrive}, \\
  c_2: \text{skibag} = \text{yes} & \rightarrow \text{type} \neq \text{city}, \\
  c_3: \text{fuel} = 41 & \rightarrow \text{type} = \text{city}, \\
  c_4: \text{fuel} = 61 & \rightarrow \text{type} \neq \text{xdrive}, \\
  c_5: \text{type} = \text{city} & \rightarrow \text{fuel} \neq 101
\end{align*}\]
Redundant Constraint (Definition)

Redundancy can be described as follows: if $C = \{c_1, c_2, \ldots, c_n\}$ is a set of constraints and one constraint $c_i \in C$ is redundant, then $(C - \{c_i\}) \cup \text{complement}(C)$ is inconsistent. In this context, \text{complement}(C) is the negation of $C$: if $C = \{c_1, c_2, \ldots, c_n\}$ then \text{complement}(C) = \{\neg c_1 \lor \neg c_2 \lor \ldots \lor \neg c_n\}$.
Definition (Redundant Constraint). Let $ca$ be a constraint of the configuration knowledge base $CKB$. $ca$ is called redundant iff $CKB - \{ca\} \nmid ca$. If this condition is not fulfilled, $ca$ is said to be nonredundant. Redundancy can also be analyzed by checking $CKB - \{ca\} \cup \text{complement}(CKB)$ for consistency. If consistency is given, $ca$ is nonredundant.
Minimal Core (Definition)

**Definition (Minimal Core).** Let $CKB$ be a configuration knowledge base. $CKB$ is denoted as minimal core iff $\forall ci \in CKB : CKB - \{ci\} \cup$ complement($CKB$) is consistent. Obviously, $CKB \cup$ complement($CKB$) $\models \bot$. 
## Algorithm 12.1 \textit{SEQUENTIAL}(C_{KB}): \Delta

\begin{align*}
\{C_{KB}: \text{configuration knowledge base}\} \\
\{C_{KB}: \text{the complement of } C_{KB}\} \\
\{\Delta: \text{set of redundant constraints}\} \\
\{C_{KBt}: \text{copy of } C_{KB} \text{ used for redundancy elimination}\} \\
C_{KBt} &\leftarrow C_{KB}; \\
\text{for all } c_i \text{ in } C_{KBt} \text{ do} \\
&\quad \text{if } \text{isInconsistent}((C_{KBt} - \{c_i\}) \cup \{-c_i\}) \text{ then} \\
&\qquad C_{KBt} \leftarrow C_{KBt} - \{c_i\}; \\
&\quad \text{end if} \\
&\text{end for} \\
\Delta &\leftarrow C_{KB} - C_{KBt}; \\
\text{return } \Delta; \\
\end{align*}
Execution Trace with SEQUENTIAL

Table 12.1  Example execution trace of SEQUENTIAL. The set of redundant constraints is $\Delta = \{c_a\}$.

<table>
<thead>
<tr>
<th>SEQUENTIAL Iteration</th>
<th>$C_{KBt}$</th>
<th>$c_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>${c_a, c_1, c_2, c_3, c_4, c_5}$</td>
<td>$c_a$</td>
</tr>
<tr>
<td>2</td>
<td>${c_1, c_2, c_3, c_4, c_5}$</td>
<td>$c_1$</td>
</tr>
<tr>
<td>3</td>
<td>${c_1, c_2, c_3, c_4, c_5}$</td>
<td>$c_2$</td>
</tr>
<tr>
<td>4</td>
<td>${c_1, c_2, c_3, c_4, c_5}$</td>
<td>$c_3$</td>
</tr>
<tr>
<td>5</td>
<td>${c_1, c_2, c_3, c_4, c_5}$</td>
<td>$c_4$</td>
</tr>
<tr>
<td>6</td>
<td>${c_1, c_2, c_3, c_4, c_5}$</td>
<td>$c_5$</td>
</tr>
</tbody>
</table>
Algorithm 12.2 CoreDiag \((C_{KB})\): \(\Delta\)

\[
\{C_{KB} = \{c_1, c_2, ..., c_n\}\}
\{C_{KB}: \text{the complement of } C_{KB}\}
\{\Delta: \text{set of redundant constraints}\}
C_{KB} \leftarrow \{\neg c_1 \lor \neg c_2 \lor ... \lor \neg c_n\};
return(C_{KB} \setminus \text{CORED}(C_{KB}, C_{KB}, C_{KB}));
\]

Algorithm 12.3 CoreD\((B, D, C = \{c_1, c_2, ..., c_p\})\): \(\Delta\)

\[
\{B: \text{consideration set}\}
\{D: \text{constraints added to } B\}
\{C: \text{set of constraints to be checked for redundancy}\}
\text{if } D \neq \emptyset \text{ and } \text{inconsistent}(B) \text{ then}
\quad return \emptyset;
\text{end if}
\text{if } \text{singleton}(C) \text{ then}
\quad return(C);
\text{end if}
\quad k \leftarrow \lceil \frac{p}{2} \rceil;
\quad C_1 \leftarrow \{c_1, c_2, ..., c_k\};
\quad C_2 \leftarrow \{c_{k+1}, c_{k+2}, ..., c_p\};
\quad \Delta_1 \leftarrow \text{CORED}(B \cup C_2, C_2, C_1);
\quad \Delta_2 \leftarrow \text{CORED}(B \cup \Delta_1, \Delta_1, C_2);
\quad return(\Delta_1 \cup \Delta_2);
\]

Redundancy Detection in Configuration Knowledge
Performance Evaluation

FIGURE 12.1
Performance of Sequential and CoreDiag for a financial services knowledge base (see Felfernig et al. 2011).
Exercises

1. Develop a redundancy-free CSP-based configuration knowledge base.

2. Include two redundant constraints.

3. Show the identification of these two redundant constraints on the basis of SEQUENTIAL.
Thank You!
References (1)


References (2)


References (3)


