Configuration Knowledge Representation and Reasoning

Lothar Hotz§, Alexander Felfernig*, Markus Stumptner†, Anna Ryabokon‡, and Claire Bagley+

*Graz University of Technology, Graz, Austria
‡University of Klagenfurt, Klagenfurt, Austria
†University of South Australia, Adelaide, Australia
+Oracle Corporate, USA
§HITeC e.V., University of Hamburg, Hamburg, Germany
Contents

• Constraint Satisfaction Problems
  • Static Constraint Satisfaction
  • Dynamic Constraint Satisfaction
  • Generative Constraint Satisfaction

• Graphical Knowledge Representations
  • Feature Models
  • UML Configuration Models
  • Formalization of UML Configuration Models
Constraint Technologies

„Constraint technologies are one of the closest approaches computer science has yet made to the Holy Grail of programming: a user states the problem, the computer solves it“

[Freuder, 1997]
Definition (Constraint Satisfaction Problem – CSP). A constraint satisfaction problem (CSP) can be defined by a triple \((V, D, C)\) where \(V\) is a set of finite domain variables \(\{v_1, v_2, \ldots, v_n\}\), \(D\) represents variable domains \(\{\text{dom}(v_1), \text{dom}(v_2), \ldots, \text{dom}(v_n)\}\), and \(C\) represents a set of constraints defining restrictions on the possible combinations of variable values \(\{c_1, c_2, \ldots, c_m\}\).
Solution for a CSP

Definition (CSP Solution). A solution for a given CSP = (V, D, C) is represented by an assignment $S = \{ins(v_1), ins(v_2), \ldots, ins(v_n)\}$ where $ins(v_i) \in dom(v_i)$. $S$ is required to be complete; that is each variable of the CSP definition has a value in $S$ and is consistent (i.e., $S$ fulfills the constraints in C).
Definition (Configuration Task). A configuration task can be defined as a CSP (V, D, C) where $V = \{v_1, v_2, \ldots, v_n\}$, $D = \{\text{dom}(v_1), \text{dom}(v_2), \ldots, \text{dom}(v_n)\}$, and $C = CKB \cup \text{REQ}$. $CKB$ represents the configuration knowledge base (the configuration model) and $\text{REQ}$ represents a set of user (customer) requirements.
Definition (Configuration). A configuration (solution) $S$ for a given configuration task $(V, D, CKB \cup REQ)$ is represented by an assignment $S = \{ins(v_1), ins(v_2), \ldots, ins(v_n)\}$ where $ins(v_i) \in dom(v_i)$ and $S$ is complete and consistent with the constraints in $CKB \cup REQ$. 
A Simple Configuration Task: Map Coloring

All regions \( y \neq x \) that are direct neighbors of \( x \) must have a different color (different from the color of \( x \)).
Corresponding Configuration Task

\[ V = \{WA, \ NT, \ SA, \ Q, \ NSW, \ V, \ T\} \]

\[ D = \{ \text{dom}(WA) = \{r,g,b\}, \ \text{dom}(NT) = \{r,g,b\}, \]
\[ \quad \text{dom}(SA) = \{r,g,b\}, \ \text{dom}(Q) = \{r,g,b\}, \ \text{dom}(NSW) = \{r,g,b\}, \]
\[ \quad \text{dom}(V) = \{r,g,b\}, \ \text{dom}(T) = \{r,g,b\}\} \]

\[ CKB = \{WA \neq NT, \ WA \neq SA, \ NT \neq SA, \ NT \neq Q, \ SA \neq Q, \]
\[ \quad SA \neq NSW, \ SA \neq V, \ Q \neq NSW, \ NSW \neq V\} \]

\[ REQ = \{WA = r\} \]
Graphical CSP Representation

FIGURE 6.2
Map coloring configuration model: graphical representation of a CSP where the nodes represent the variables and the arcs represent constraints.
Graphical CSP Representation

**FIGURE 6.3**
Simple Mobile Phone configuration model (represented as CSP). An abbreviation is used for the constraint representation; for example, $CA \leftrightarrow MP$ is the short form of $CA = 1 \leftrightarrow MP = 1$. $MP =$ Mobile Phone, $CA =$ Calls, $GPS =$ GPS, $SC =$ Screen, $CM =$ Camera, $MPX =$ MPX Player, $BA =$ Basic, $CO =$ Color, $HR =$ High Resolution.
CSP Solution Search: Forward Checking

**FIGURE 6.4**
A simple example of forward checking. The variables WA and NT already have assigned values (r and g). The only possible remaining value for SA is b; r and g do not have to be checked for consistency with the settings of WA and NT.
Dynamic Constraint Satisfaction

- Reasoning over variables states
- Only active variables are part of the solution
- Activation constraints determine activity status of a variable
- HighResolution (Camera) = yes → active(HighResolution).

[Mittal and Falkenhainer, 1990]
Generative Constraint Satisfaction (GCSP)

- Representational limits of discussed approaches
- Component-oriented representation not possible (only variables and constraints)
- Not applicable if number of components depends on the preferences of a user
- Need for “on the fly“ generation of components
Generative Constraint Satisfaction (GCSP)

**PC P DESCRIPTION:**
P.name := [String];
P.price := [Integer];
P.usage := {'internet','scientific','multimedia'};
P.efficiency := {'A','B','C'};
P.PORTS := {screen-of-pc-1[Screen], screen-of-pc-2[Screen],
             hdunit-of-pc-1[HDUnit], ...};
P.screens := <screen-of-pc-1,screen-of-pc-2>;
P.mb := <mb-of-pc-1>;
P.hdunits := <hdunit-of-pc-1,hdunit-of-pc-2>;
P.efficiency = P.mb.efficiency; /* Example constraint*/
Solution Search (GCSP)

**FIGURE 6.5**
Simple GCSP dynamic component creation example. The constraint representation is simplified (i.e., it does not directly correspond to the GCSP constraint representation used in Stumptner et al., 1998).
Graphical Knowledge Representations

- Need to improve maintainability of configuration models
- Approach: graphical knowledge representations
- Automated translation into executable representation
- Examples:
  - Feature Models
  - UML Configuration Models
Feature Models

**FIGURE 6.6**
Feature model of a mobile phone (adapted version of Benavides et al., 2010). This feature model is equivalent to the constraint-based representation of Figure 6.3.
## Semantics of Feature Models

Table 6.1 Semantics of feature models defined in terms of constraints (propositional logic). Features are represented by Boolean variables.

<table>
<thead>
<tr>
<th>Relationship/Constraint</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>mandatory(P,C)</td>
<td>$P \iff C$</td>
</tr>
<tr>
<td>optional(P,C)</td>
<td>$C \rightarrow P$</td>
</tr>
<tr>
<td>or $(P, C_1, C_2, \ldots, C_n)$</td>
<td>$P \leftrightarrow (C_1 \lor C_2 \lor \ldots \lor C_n)$</td>
</tr>
<tr>
<td>alternative $(P, C_1, C_2, \ldots, C_n)$</td>
<td>$(C_1 \leftrightarrow (\neg C_2 \land \ldots \land \neg C_n \land P)) \land (C_2 \leftrightarrow (\neg C_1 \land \neg C_3 \land \ldots \land \neg C_n \land P)) \land \ldots \land (C_n \leftrightarrow (\neg C_1 \land \ldots \land \neg C_{n-1} \land P))$</td>
</tr>
<tr>
<td>requires(P,C)</td>
<td>$P \rightarrow C$</td>
</tr>
<tr>
<td>excludes(P,C)</td>
<td>$\neg P \lor \neg C$</td>
</tr>
</tbody>
</table>
UML Configuration Model

**FIGURE 6.9**
Fragment of the PC model (adapted part of Figure 6.7).
### UML Configuration Model: Constraints

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gc1</td>
<td>CPUs of type CPUS are incompatible with motherboards of type MBDiamond</td>
</tr>
<tr>
<td>gc2</td>
<td>CPUs of type CPUD are incompatible with motherboards of type MBSilver</td>
</tr>
<tr>
<td>gc3</td>
<td>Each OS of type OSAalpha requires a CPU of type CPUD</td>
</tr>
<tr>
<td>prc2’</td>
<td>The price of one personal computer (PC) is determined by the prices of the motherboard (MB), the CPUs, and the operating system (OS)</td>
</tr>
<tr>
<td>resc1</td>
<td>The computer price must be less or equal to the maxprice defined by the customer</td>
</tr>
</tbody>
</table>
### UML Configuration Model: Formalization of Product Structure

**Table 6.4** Example formalizations of the model \( (C_{KB}) \) depicted in Figure 6.9. \( getcpus \) denotes a collection operator (Felfernig et al., 2000a) that collects all \( cpus \) connected with motherboard \( Y \). For reasons of readability we limit the example to attribute range restrictions (e.g., PC\( (\text{efficiency}) \)).

<table>
<thead>
<tr>
<th>Modeling Facility</th>
<th>Example in FOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component types</td>
<td>{PC/1, MB/1, MBDiamond/1, MBSilver/1, CPU/1, CPUS/1, CPUD/1, OS/1, OSAIpha/1, OSBeta/1} ( C_{LANG} )</td>
</tr>
<tr>
<td>Attributes</td>
<td>{efficiency/2, price/2, maxprice/2, clockrate/2, hdcapacity/2} ( C_{LANG} )</td>
</tr>
<tr>
<td>Relationships</td>
<td>{pc-of-mb/2, mb-of-pc/2, mb-of-cpu/2, cpu-of-mb/2, pc-of-os/2, os-of-pc/2} ( C_{LANG} )</td>
</tr>
<tr>
<td>PC (efficiency)</td>
<td>( \forall X : PC(X) \rightarrow \exists Y : \text{efficiency}(X, A_X) \land A_X = A \lor A_X = B \lor A_X = C \rightarrow C_{KB} )</td>
</tr>
<tr>
<td>MB (efficiency)</td>
<td>( \forall X : MB(X) \rightarrow \exists Y : \text{efficiency}(X, A_X) \land A_X = A \lor A_X = B \lor A_X = C \rightarrow C_{KB} )</td>
</tr>
<tr>
<td>MB (price)</td>
<td>( \forall X : MB(X) \rightarrow \exists Y : \text{price}(X, A_X) \land A_X \geq 0 \lor A_X \leq 350 \rightarrow C_{KB} )</td>
</tr>
<tr>
<td>CPUS (price)</td>
<td>( \forall X : CPUS(X) \rightarrow \exists Y : \text{price}(X, A_X) \land A_X = 100 \rightarrow C_{KB} )</td>
</tr>
<tr>
<td>part-of (PC,MB)</td>
<td>( \forall X : PC(X) \rightarrow \exists Y : MB(Y) \land \text{pc-of-mb}(X, Y) \rightarrow C_{KB} )</td>
</tr>
<tr>
<td></td>
<td>( \forall X : MB(X) \rightarrow \exists Y : PC(Y) \land \text{mb-of-pc}(X, Y) \rightarrow C_{KB} )</td>
</tr>
<tr>
<td>part-of (PC,OS)</td>
<td>( \forall X : PC(X) \rightarrow \exists Y : OS(Y) \land \text{pc-of-os}(X, Y) \rightarrow C_{KB} )</td>
</tr>
<tr>
<td></td>
<td>( \forall X : OS(X) \rightarrow \exists Y : PC(Y) \land \text{os-of-pc}(X, Y) \rightarrow C_{KB} )</td>
</tr>
</tbody>
</table>
UML Configuration Model: Formalization of Constraints

\[
\begin{align*}
\text{gc1} & \quad \{ \forall X, Y : \text{mb-of-cpu}(X, Y) \land \text{MBDiamond}(X) \land \text{CPUS}(Y) \rightarrow \text{false.}) \subseteq C_{KB} \\
\text{gc2} & \quad \{ \forall X, Y : \text{mb-of-cpu}(X, Y) \land \text{MBSilver}(X) \land \text{CPUD}(Y) \rightarrow \text{false.}) \subseteq C_{KB} \\
\text{gc3} & \quad \{ \forall X, Y : \text{PC}(X) \land \text{OSAlpha}(Y) \land \\
& \quad \quad \quad \text{pc-of-os}(X, Y) \rightarrow \exists U, V : \text{MB}(U) \land \text{CPUD}(V) \land \text{pc-of-mb}(X, U) \land \\
& \quad \quad \quad \text{mb-of-cpu}(U, V).) \subseteq C_{KB} \\
\text{prc2'} & \quad \{ \forall X : \text{PC}(X) \land \text{price}(X, PCP) \land \text{pc-of-mb}(X, Y) \land \\
& \quad \quad \quad \text{pc-of-os}(X, Z) \land \text{getcpus}(Y, CPUs) \rightarrow \text{PCP} = \\
& \quad \quad \quad \sum_{c \in \{Y, Z\} \cup \text{CPUs} \land \text{price}(c, P).} P. \subseteq C_{KB} \\
\text{resc1} & \quad \{ \forall X : \text{PC}(X) \land \text{price}(X, PCP) \land \text{maxprice}(X, PCMP) \rightarrow \text{PCP} \leq \text{PCMP}.) \subseteq C_{KB}
\end{align*}
\]
Exercises

1. Explain the concept of forward checking on the basis of an example.

2. Translate the Mobile Phone feature model into a corresponding CSP-based representation.

3. Implement the Mobile Phone feature model with the CHOCO constraint solver (http://choco-solver.org)

4. Develop a feature model for a product domain of your own choice (not discussed in lecture).

5. Translate the following UML Model (next slide) into a logic-based representation.
Exercises (UML Model)
Thank You!
References (1)


References (2)


References (3)


References (4)


References (6)


References (7)


References (8)


References (9)


References (10)


References (11)


References (12)


References (13)


References (14)


References (15)

