# Conflict Detection and Diagnosis in Configuration

Alexander Felfernig[*], Stefan Reiterer[†], Florian Reinfrank[*], Gerald Ninaus[*], and Michael Jeran[*]

[*]Graz University of Technology, Graz, Austria
[†]SelectionArts, Graz, Austria

# Contents

- Anomalies
- Conflict Detection
  - Simple Conflict Detection
  - QuickXPlain
- Diagnosis
  - HSDAG
  - Duality of Conflicts and Diagnoses
  - FastDiag

# Anomalies

- Parts of a knowledge base that conform to a defined pattern of unintended structures [Chandola et al., 2009].

- Anomaly detection:
  - Automated testing and debugging (minimal subsets)
  - Redundancy detection (maximal subsets)

# Working Example: Knowledge Base

$C_{KB} = \{$

$c_\alpha : \forall X : MB(X) \rightarrow \exists_1^1 Y :$ cpu-of-mb(Y, X).

$c_\beta : \forall X : CPU(X) \rightarrow \exists_1^1 Y :$ mb-of-cpu(Y, X).

$c_\gamma : \forall X : MB(X) \leftrightarrow MBSilver(X) \vee MBDiamond(X).$

$c_\delta : \forall X : \neg MBSilver(X) \vee \neg MBDiamond(X).$

$c_\epsilon : \forall X : CPU(X) \leftrightarrow CPUD(X) \vee CPUS(X).$

$c_\phi : \forall X, Y :$ cpu-of-mb(X,Y) $\leftrightarrow$ mb-of-cpu(Y,X).

$c_\iota : \forall X : \neg CPUD(X) \vee \neg CPUS(X).$

$c_1 : \forall X, Y :$ cpu-of-mb (Y, X) $\wedge\ CPUS(Y) \rightarrow MBDiamond(X).$

$c_2 : \forall X, Y :$ cpu-of-mb $(Y, X) \wedge CPUS(Y) \rightarrow MBSilver(X).$ **/*should be: replacement of c1*/**

$c_3 : \forall X, Y :$ cpu-of-mb $(Y, X) \wedge CPUD(Y) \wedge MBSilver(X) \rightarrow false.$

$c_4 : \forall X, Y :$ cpu-of-mb $(Y, X) \wedge CPUS(Y) \wedge MBDiamond(X) \rightarrow false.$

$c_5 : \forall X : CPUD(X) \rightarrow false.$ **/*should be disabled, but still active*/**

$\}$

# Minimal Conflicts

**Definition (Minimal Conflict Set).** A conflict set CS = {*ca*, *cb*, . . ., *cz*} is a subset of *C* such that *inconsistent* (*B* ∪ *CS*). *AC* = *B* ∪ *C* represents the set of all constraints in the knowledge base (*AC* = {*c*1, *c*2, . . ., *cn*}), *B* represents the background knowledge (no conflict elements are assumed to be included in *B*), and *C* represents the set of constraints subject of conflict search. A conflict set CS is *minimal* if there does not exist a *CS'* ⊂ *CS* that has the conflict property.

# Minimal Conflicts

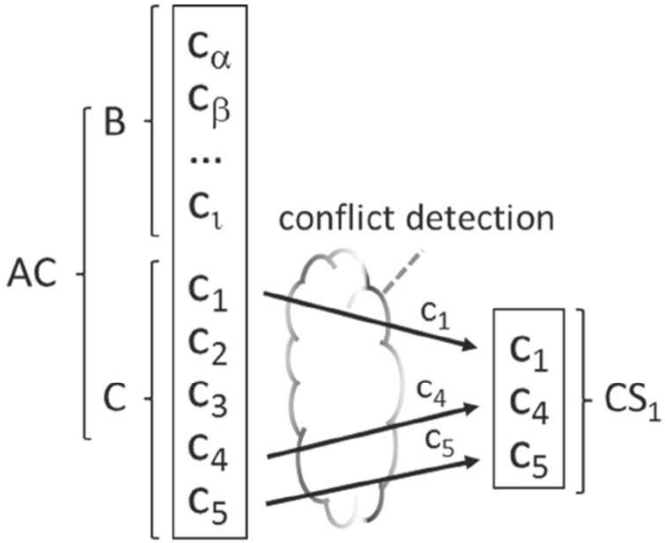


**FIGURE 7.1**

A conflict set CS is a subset of $C$ ($AC = C \cup B$), which is inconsistent with B. CS is minimal if no subset of CS fulfills the conflict set property. In this context, B is the background knowledge that includes all constraints considered correct. An example conflict set is $CS_1 = \{c_1, c_4, c_5\}$.

# Working Example: Minimal Conflicts

- Assumption: C = AC, i.e., B = {}
- $CS1 = \{c1, c4, c5\}$
- $CS2 = \{c1, c2, c5\}$
- $inconsistent(\{c1, c4, c5\} \cup B)$
- $inconsistent\ (\{c1, c2, c5\} \cup B)$
- Two algorithms to determine minimal conflicts:
  - Simple Conflict Detection
  - QuickXPlain

# Simple Conflict Detection

**Algorithm 7.1** – SIMPLECONFLICTDETECTION

```
1  func SimpleConflictDetection(C ⊆ AC, B = AC − C}) : CS
2  CS ← ∅;
3  if inconsistent(B) or consistent(B ∪ C) return(∅);
4  else
5  repeat
6    Φ = CS;
7    repeat
8      c ← element(C − Φ);
9      Φ ← Φ ∪ {c};
10   until inconsistent(Φ)
11   CS ← CS ∪ {c};
12 until inconsistent(CS)
13 return(CS);
```

**Best Case**:

$2$

**Worst Case**:

$$\frac{(n \times (n+1))}{2} + n$$

# Simple Conflict Detection: Execution

**Table 7.1** Example of the application of SIMPLECONFLICTDETECTION. $CS = \{c_1, c_4, c_5\}$ is returned as minimal conflict set (CS) for $C = \{c_5, c_4, c_3, c_2, c_1\}$ and $B = \{c_\alpha, c_\beta, c_\gamma, c_\delta, c_\epsilon, c_\phi, c_\iota\}$.

| Step | CS | c | Φ |
|---|---|---|---|
| 1 | ∅ | $c_5$ | $\{c_5\}$ |
| 2 | ∅ | $c_4$ | $\{c_5, c_4\}$ |
| 3 | ∅ | $c_3$ | $\{c_5, c_4, c_3\}$ |
| 4 | ∅ | $c_2$ | $\{c_5, c_4, c_3, c_2\}$ |
| 5 | ∅ | $c_1$ | $\{c_5, c_4, c_3, c_2, c_1\}$ |
| 6 | $\{c_1\}$ | $c_5$ | $\{c_1, c_5\}$ |
| 7 | $\{c_1\}$ | $c_4$ | $\{c_1, c_5, c_4\}$ |
| 8 | $\{c_1, c_4\}$ | $c_5$ | $\{c_1, c_4, c_5\}$ |
| 9 | $\{c_1, c_4, c_5\}$ | – | – |

# QuickXPlain

**Algorithm 7.2 – QuickXPlain**

1 **func** QuickXPlain$(C \subseteq AC, B = AC - C) : CS$
2 **if** $isEmpty(C)$ *or* $inconsistent(B)$ *return* $\emptyset$;
3 **else** *return* $\mathrm{QX}(\emptyset, C, B)$;

4 **func** $\mathrm{QX}(D, C = \{c_1..c_q\}, B) : CS$
5 **if** $D \neq \emptyset$ *and* $inconsistent(B)$ *return* $\emptyset$;
6 **if** $singleton(C)$ *return* $C$;
7 $k = \lceil \frac{q}{2} \rceil$;
8 $C_1 = \{c_1..c_k\}; C_2 = \{c_{k+1}..c_q\}$;
9 $CS_1 = \mathrm{QX}(C_2, C_1, B \cup C_2)$;
10 $CS_2 = \mathrm{QX}(CS_1, C_2, B \cup CS_1)$;
11 $return(CS_1 \cup CS_2)$;

**Best Case**:
$log_2\left(\frac{n}{k}\right)+2k$

**Worst Case**:
$2k \times log_2\left(\frac{n}{k}\right)+2k$

# QuickXPlain

**Table 7.2** Example of QUICKXPLAIN: $\Gamma = \{c_\alpha, c_\beta, c_\gamma, c_\delta, c_\epsilon, c_\phi, c_\iota\}$ is the (original) background knowledge and CS $= \{c_1, c_4, c_5\}$ is the returned conflict set. The sequence of the different QX activations is depicted in Figure 7.2.

| Step | D | C | B | $C_1$ | $C_2$ | Return |
|---|---|---|---|---|---|---|
| 1 | $\emptyset$ | $\{c_1, \ldots, c_5\}$ | $\Gamma$ | $\{c_1, c_2, c_3\}$ | $\{c_4, c_5\}$ | $\{c_1, c_4, c_5\}$ |
| 2 | $\{c_4, c_5\}$ | $\{c_1, c_2, c_3\}$ | $\Gamma \cup \{c_4, c_5\}$ | $\{c_1, c_2\}$ | $\{c_3\}$ | $\{c_1\}$ |
| 3 | $\{c_3\}$ | $\{c_1, c_2\}$ | $\Gamma \cup \{c_3, \ldots, c_5\}$ | $\{c_1\}$ | $\{c_2\}$ | $\{c_1\}$ |
| 4 | $\{c_2\}$ | $\{c_1\}$ | $\Gamma \cup \{c_2, \ldots, c_5\}$ | $\emptyset$ | $\emptyset$ | $\{c_1\}$ |
| 5 | $\{c_1\}$ | $\{c_2\}$ | $\Gamma \cup \{c_1, c_3, c_4, c_5\}$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| 6 | $\{c_1\}$ | $\{c_3\}$ | $\Gamma \cup \{c_1, c_4, c_5\}$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| 7 | $\{c_1\}$ | $\{c_4, c_5\}$ | $\Gamma \cup \{c_1\}$ | $\{c_4\}$ | $\{c_5\}$ | $\{c_4, c_5\}$ |
| 8 | $\{c_5\}$ | $\{c_4\}$ | $\Gamma \cup \{c_1, c_5\}$ | $\emptyset$ | $\emptyset$ | $\{c_4\}$ |
| 9 | $\{c_4\}$ | $\{c_5\}$ | $\Gamma \cup \{c_1, c_4\}$ | $\emptyset$ | $\emptyset$ | $\{c_5\}$ |

# Runtime of Conflict Detection Algorithms

**Table 7.3** Runtime evaluation: The average runtime in milliseconds (ms) needed by SIMPLECONFLICTDETECTION (SCD) and QUICKXPLAIN to calculate one minimal conflict set (on a standard PC). The basis for this evaluation are knowledge bases from www.splot-research.org.

| Domain | #con. | #var. | QUICKXPLAIN (ms) | SCD (ms) |
|---|---|---|---|---|
| DELL laptops | 285 | 47 | 75.7 | 643.2 |
| Smarthomes | 73 | 55 | 42.6 | 89.6 |
| Cars | 150 | 73 | 42.9 | 406.2 |
| Xerox printers | 242 | 158 | 78.1 | 812.2 |

# Diagnosis (Task)

**Definition (Diagnosis Task).** A *diagnosis task* can be defined by the tuple (*C, AC*) where *AC* = *B* ∪ *C*, *B* is the background knowledge, and *C* is the set of constraints to be analyzed.

**Definition (Diagnosis).** A *diagnosis* for a given diagnosis task (C,AC) is a set of constraints $\Delta \subseteq C$ such that $B \cup C - \Delta$ is consistent. A diagnosis $\Delta$ is *minimal* if there does not exist a diagnosis $\Delta' \subset \Delta$ with the diagnosis property. Finally, a minimal diagnosis $\Delta$ is denoted as *minimal cardinality diagnosis* if there does not exist a minimal diagnosis with $|\Delta'| < |\Delta|$.

# Minimal Diagnoses

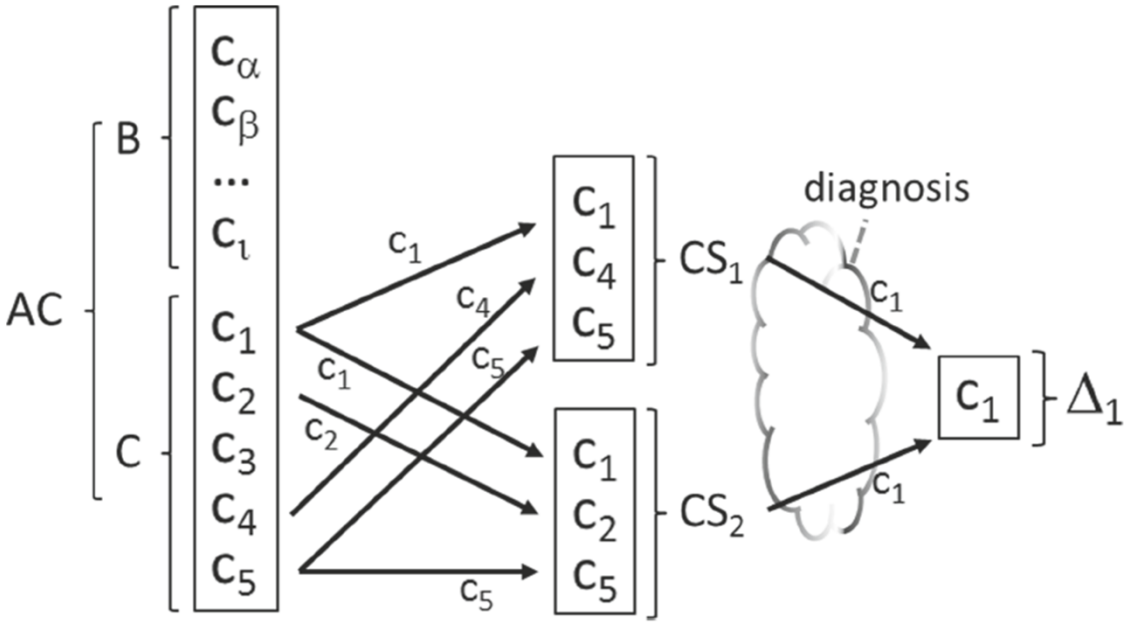


**FIGURE 7.3**

A diagnosis $\Delta$ is a subset of $C$ ($AC = C \cup B$) such that $B \cup C - \Delta$ is consistent. $\Delta$ is minimal if no subset of $\Delta$ fulfills the diagnosis property. B again represents the background knowledge. An example diagnosis is $\Delta_1 = \{c_1\}$.

# Hitting Set Directed Acyclic Graph (HSDAG)

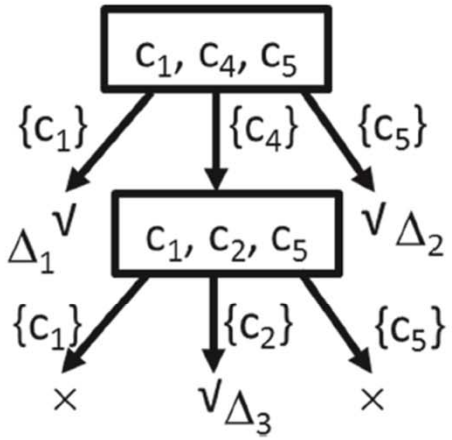


**FIGURE 7.4**

Breadth-first based search for diagnoses on the basis of the minimal conflict sets $CS_1 = \{c_1, c_4, c_5\}$ and $CS_2 = \{c_1, c_2, c_5\}$. The resulting minimal diagnoses are $\Delta_1 = \{c_1\}$, $\Delta_2 = \{c_5\}$, and $\Delta_3 = \{c_2, c_4\}$.

# Duality of Conflicts and Diagnoses

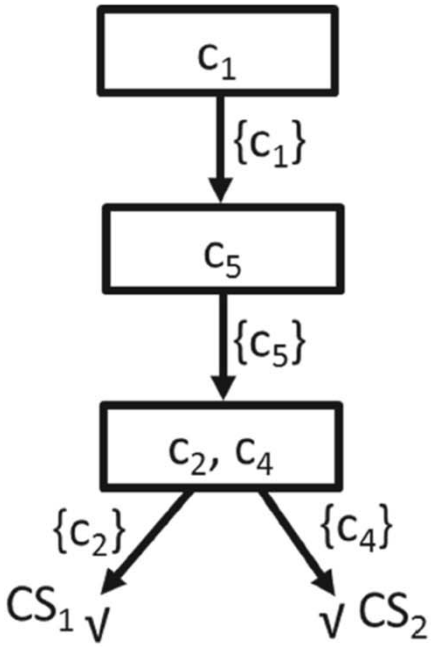


**FIGURE 7.5**

Breadth-first based search for conflicts on the basis of the minimal diagnoses $\Delta_1 = \{c_1\}$, $\Delta_2 = \{c_5\}$, and $\Delta_3 = \{c_2, c_4\}$. The resulting minimal conflict sets are $CS_1 = \{c_1, c_2, c_5\}$, $CS_2 = \{c_1, c_4, c_5\}$.

# FastDiag

**Algorithm 7.3 – FastDiag**

1 **func** FastDiag($C \subseteq AC, AC = \{c_1..c_t\}$) : *diagnosis* $\Delta$
2 **if** *isEmpty*($C$) *or inconsistent*($AC - C$) *return* $\emptyset$
3 **else** *return* FD($\emptyset, C, AC$);

4 **func** FD($D, C = \{c_1..c_q\}, AC$) : *diagnosis* $\Delta$
5 **if** $D \neq \emptyset$ *and consistent*($AC$) *return* $\emptyset$;
6 **if** *singleton*($C$) *return* $C$;
7 $k = \frac{q}{2}$;
8 $C_1 = \{c_1..c_k\}; C_2 = \{c_{k+1}..c_q\}$;
9 $D_1 = \text{FD}(C_2, C_1, AC - C_2)$;
10 $D_2 = \text{FD}(D_1, C_2, AC - D_1)$;
11 *return*$(D_1 \cup D_2)$;

**Best Case**:
$log_2\left(\frac{n}{d}\right) + 2d$

**Worst Case**:
$2d \times log_2\left(\frac{n}{d}\right) + 2d$

# Runtime of Diagnosis Algorithms

**Table 7.5** Runtime evaluation: The average runtime in milliseconds (ms) needed by HSDAG and FASTDIAG to calculate one minimal diagnosis (on a standard PC). The basis for this evaluation are knowledge bases from www.splot-research.org (Dell laptops (laptops), smarthomes (homes), cars, and Xerox printers (printers)).

| Domain | #con. | #var. | FASTDIAG (ms) | | | HSDAG (ms) | | |
|---|---|---|---|---|---|---|---|---|
| $\#\Delta_i$ | | | 1 | 5 | 10 | 1 | 5 | 10 |
| Laptops | 285 | 47 | 1638.7 | 2792.3 | 3464.1 | 2668.9 | 4977.6 | 5336.3 |
| Homes | 73 | 55 | 593.1 | 2433.5 | 3167.8 | 2074.2 | 2151.4 | 2241.2 |
| Cars | 150 | 73 | 1404.4 | 2730.8 | 3606.0 | 5741.1 | 6347.9 | 6942.0 |
| Printers | 242 | 158 | 2871.9 | 6927.2 | 12091.0 | >100k | >100k | >100k |

# Diagnosis as Optimization Problem

- Each constraint ci is represented by a variable xi [0,1]
- Conflict sets CSj are represented by constraints csj
- Example: cs1: x1 + x4 + x5 ≥ 1

$$cs_1 : x_1 + x_4 + x_5 \geq 1.$$
$$cs_2 : x_1 + x_2 \geq 1.$$
$$cs_3 : x_1 + x_3 \geq 1.$$
$$cs_4 : x_2 + x_3 \geq 1.$$

- Optimization function:

$$minimize : x_1 + x_2 + x_3 + x_4 + x_5$$

# Diagnosis as Optimization Problem

**Table 7.6** Representation of a diagnosis task as optimization problem. In this case, all minimal conflict sets ($CS_1, \ldots, CS_4$) have to be determined before the optimization can start (1 (0) denotes the fact that $c_i$ is part (not part) of the minimal conflict set).

| Conflict Set | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ |
|---|---|---|---|---|---|
| $CS_1$ | 1 | 0 | 0 | 1 | 1 |
| $CS_2$ | 1 | 1 | 0 | 0 | 0 |
| $CS_3$ | 1 | 0 | 1 | 0 | 0 |
| $CS_4$ | 0 | 1 | 1 | 0 | 0 |

**cs1: x1 + x4 + x5 ≥ 1**

# Solution Space: Predefined Instances

**Table 7.7** A simple configuration problem defined by the variables $V = \{v_1, v_2, v_3\}$, $\text{dom}(v_i) = \{1,2,3\}$, and the constraint $c_p = conf_1 \vee conf_2 \vee conf_3 \vee conf_4 \in C_{KB}$.

| Variables | $conf_1$ | $conf_2$ | $conf_3$ | $conf_4$ |
|---|---|---|---|---|
| $v_1$ | 1 | 3 | 1 | 1 |
| $v_2$ | 2 | 2 | 2 | 2 |
| $v_3$ | 3 | 1 | 2 | 1 |

**Table 7.8** Example user requirements $C_R$ and their relationship to the configurations $conf_1, \ldots, conf_4$ (1 = requirement supported, 0 = not supported).

| User Requirements | $conf_1$ | $conf_2$ | $conf_3$ | $conf_4$ |
|---|---|---|---|---|
| $v_1 = 1$ | 1 | 0 | 1 | 1 |
| $v_2 = 1$ | 0 | 0 | 0 | 0 |
| $v_3 = 1$ | 0 | 1 | 0 | 1 |
| *support* | 1 | 1 | 1 | 2 |

# Exercises

1. Given the following set of constraints AC={x1=1, x1=2, x2=x1, x3=x2, x3>2} (dom(xi)=[1,2,3]). Determine the complete set of minimal conflicts on the basis of Simple Conflict Detection.
2. For the identified minimal conflict sets determine the corresponding complete set of minimal diagnoses (on the basis of HSDAG).
3. Show how to use the HSDAG concept to determine the complete set of minimal conflicts from the diagnoses determined in 2 (duality of diagnoses and conflicts).
4. For the minimal conflict sets (from 1. and 3.), show how to represent a diagnosis as an optimization problem.

# Thank You!

# References (1)


(1) Barker, V., O'Connor, D., Bachant, J., Soloway, E., 1989. Expert systems for configuration at Digital: XCON and beyond. Communications of the ACM 32 (3), 298–318.

(2) Chandola, V., Banerjee, A., Kumar, V., 2009. Anomaly Detection: A Survey. ACM Computing Surveys 41 (3), 1–58.

(3) Felfernig, A., Friedrich,G., Jannach,D., Stumptner, M., 2004. Consistency-based diagnosis of configuration knowledge bases. Artificial Intelligence 152 (2), 213–234.

(4) Felfernig, A., Zehentner, C., Blazek, P., 2011. CoreDiag: eliminating redundancy in constraint sets. In: 22nd International Workshop on Principles of Diagnosis (DX'2011), Murnau, Germany, pp. 219–224.

(5) Felfernig, A., Schubert, M., Zehentner, C., 2012. An efficient diagnosis algorithm for inconsistent constraint sets. Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM) 26 (1), 53–62.

(6) Felfernig, A., Schippel, S., Leitner, G., Reinfrank, F., Isak, K., Mandl,M., Blazek, P., Ninaus, G., 2013. Automated repair of scoring rules in constraint-based recommender systems. AI Communications 26 (2), 15–27.

# References (2)


(7) Felfernig, A., Reinfrank, F., Ninaus, G., Blazek, P., 2014. Redundancy detection in configuration knowledge. In: Felfernig, A., Hotz, L., Bagley, C., Tiihonen, J. (Eds.), Knowledge-based Configuration – From Research to Business Cases. Morgan Kaufmann Publishers, Waltham, MA, pp. 157–166 (Chapter 12).

(8) Fijany, A., Vatan, F., 2004. New approaches for efficient solution of hitting set problem. In: Winter International Symposium on Information and Communication Technologies. Trinity College Dublin, Cancun, Mexico, pp. 1–6.

(9) Fleischanderl, G., Friedrich, G.E., Haselböck, A., Schreiner, H., Stumptner, M., 1998. Configuring large systems using generative constraint satisfaction. IEEE Intelligent Systems 13 (4), 59–68.

(10) Friedrich, G., Jannach, D., Stumptner, M., Zanker, M., 2014. Knowledge engineering for configuration systems. In: Felfernig, A., Hotz, L., Bagley, C., Tiihonen, J. (Eds.), Knowledge-based Configuration – From Research to Business Cases. Morgan Kaufmann, pp. 139–155 (Chapter 11).

(11) Hotz, L., Felfernig, A., Stumptner, M., Ryabokon, A., Bagley, C., Wolter, K., 2014. Configuration knowledge representation and reasoning. In: Felfernig, A., Hotz, L., Bagley, C., Tiihonen, J. (Eds.), Knowledge-based Configuration – From Research to Business Cases. Morgan Kaufmann, pp. 41–72 (Chapter 6).

# References (3)


(12)Jannach,D., 2008. Finding preferred query relaxations in content-based recommenders. In: Chountas, P., Petrounias, I., Kacprzyk, J. (Eds.), Intelligent Techniques and Tools for Novel System Architectures, London, UK, Studies in Computational Intelligence, vol. 109, pp. 81–97.

(13)Junker, U., 2004. QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems. In: McGuinness, D.L., Ferguson, G. (Eds.), 19th International Conference on Artificial Intelligence (AAAI'04). AAAI Press, San Jose, CA, pp. 167–172.

(14)Reiter, R., 1987. A theory of diagnosis from first principles. Artificial Intelligence 32 (1), 57–95.

(15)Schubert,M., Felfernig, A., 2011. BFX: diagnosing conflicting requirements in constraint-based recommendation. International Journal on Artificial Intelligence Tools 20 (2), 297–312.

(16)Schubert, M., Felfernig, A., Mandl, M., 2010. FastXPlain: conflict detection for constraint-based recommendation problems. In: García-Pedrajas, N., Herrera, F., Fyfe, C., Benítez, J., Ali, M. (Eds.), IEA/AIE 2010. Lecture Notes in Computer Science, vol. 6096. Springer, Cordoba, Spain, pp. 621–630.