Maintaining Constraint-based Configuration Systems: Challenges ahead

Florian Reinfrank, Gerald Ninaus, Franz Wotawa, Alexander Felfernig

{firstname.lastname}@ist.tugraz.at

Institute for Software Technology

Graz University of Technology

Inffeldgasse 16b, 8010 Graz, Austria

Outline

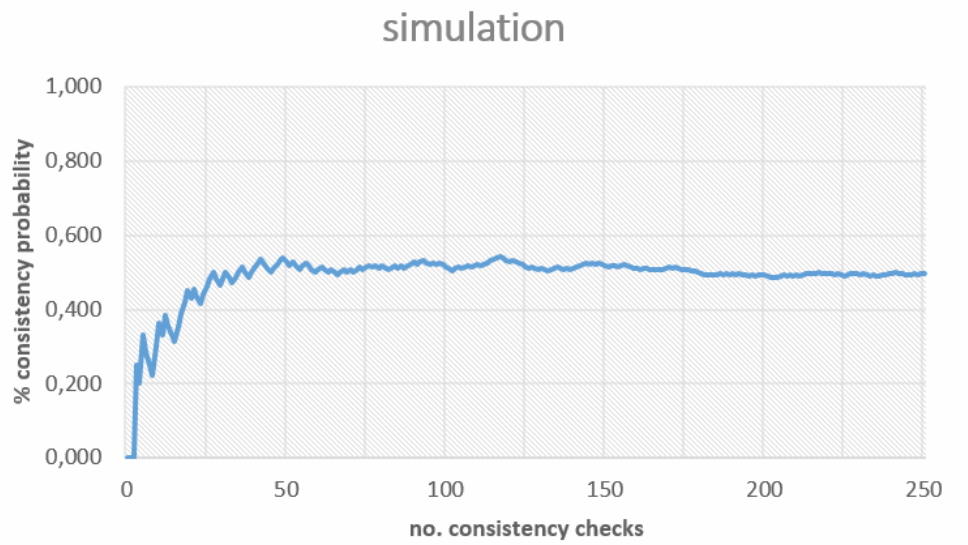
- Introduction
- Simulation
- Gibbs sampling
- Advantages
- Summary

Introduction

- Constraint-based configuration knowledge bases are complex
  - Hundreds of variables
  - Thousands of constraints
  - Billions of consistent variants
- Requirements for configuration knowledge bases
  - Evaluate the configuration knowledge base
  - Testing the knowledge base (is each consistent variant correct?)
  - Market analysis (do the consistent variants cover the market needs?)
  - Guide the customer through the product space (avoid the no-solution-could-be-found dilemma)

Simulation

- Approximate the number of consistent variants compared to all variants
- Add a set of randomly generated assignments
- Check if the set is consistent



Simulation: Gibbs sampling

Function $Gibbs(KB)$
- $n = 0$
- $mincalls = 200$
- $threshold = 0.01$
- $verify = Double.maxValue$
- $CC = \emptyset$ // *consistencychecks*
- ***While*** $n < mincalls \vee verify > threshold$
  - ***If*** $isConsistent(\underline{genRandomAssignments}(KB))$
    - $CC.add(1)$
  - ***Else***
    - $CC.add(0)$
  - ***EndIf***
  - $verify = \underline{verifyChecks}(CC)$
  - $n{+}{+}$
- ***EndWhile***
- ***Return*** $(consistent(CC)/|CC|)$

EndFunction

Simulation: Gibbs sampling

```
Function Gibbs(KB)
    n = 0
    mincalls = 200
    threshold = 0.01
    verify = Double.maxValue
    CC = Ø // consistencychecks
    While n < mincalls ∨ verify > threshold
        If isConsistent(genRandomAssignments(KB))
            CC.add(1)
        Else
            CC.add(0)
        EndIf
        verify = verifyChecks(CC)
        n++
    EndWhile
    Return (consistent(CC)/|CC|)
EndFunction
```

```
Function genRandomAssignments(KB)
    n = Random(1, |V ∈ KB|)
    KB' = KB
    Do n Times
        v = selectRandom(V ∈ KB)
        r = selectRandom({<, ≤, =, ≠, ≥, >})
        d = selectRandom(dom(v))
        KB'.addConstraint(
            newConstraint(v, r, d))
    EndDo
    Return KB'
EndFunction
```

Simulation: Gibbs sampling

```
Function Gibbs(KB)
    n = 0
    mincalls = 200
    threshold = 0.01
    verify = Double.maxValue
    CC = Ø // consistencychecks
    While n<mincalls ∨ verify>threshold
        If isConsistent(genRandomAssignments(KB))
            CC.add(1)
        Else
            CC.add(0)
        EndIf
        verify = verifyChecks(CC)
        n++
    EndWhile
    Return(consistent(CC)/|CC|)
EndFunction
```

```
Function genRandomAssignments(KB)
    n = Random(1,|V∈KB|)
    KB' = KB
    Do n Times
        v = selectRandom(V∈KB)
        r = selectRandom({<,≤,=,≠,≥,>})
        d = selectRandom(dom(v))
        KB'.addConstraint(
            newConstraint(v,r,d))
    EndDo
    Return KB'
EndFunction
```

```
Function verifyChecks(CC)
    CC_1 = {cc_0, cc_1, ..., cc_n/2}
    CC_2 = {cc_n/2+1, cc_n/2+2, ..., cc_n}
    return √((mean(CC_1) − mean(CC_2))^2)
EndFunction
```

Advantages: restriction rate

- Estimate the coverage

$$coverage(C) = \frac{number\ of\ consistent\ variants}{number\ of\ all\ possible\ variants}$$

$$0 \leq coverage(C) \leq 1$$

- Using for knowledge base evaluations

Advantages: test case generation

- Boundary value analysis
- Generate random assignments
- Get the coverage of these assignments

Advantages: test case generation

- Boundary value analysis
- Generate random assignments
- Get the coverage of these assignments

test case	*filter constraint*	*coverage*
t_0	*cpuCores = 2 ∧ usageScenario = office*	*0.50*
t_1	*cpuCores = 2 ∧ usageScenario = multimedia*	*0.50*
t_2	*price = 799 ∧ usageScenario = gaming*	*0.00*
t_3	*price = 599 ∧ usageScenario = gaming*	*0.50*
t_4	*cpuCores = 4 ∧ usageScenario = multimedia*	*0.50*
t_5	*cpuCores = 4*	*~0.54*

Advantages: test case generation

- Boundary value analysis
- Generate random assignments
- Get the coverage of these assignments

test case	*filter constraint*	*coverage*
t_0	*cpuCores = 2 ∧ usageScenario = office*	*0.50*
t_1	*cpuCores = 2 ∧ usageScenario = multimedia*	*0.50*
t_2	*price = 799 ∧ usageScenario = gaming*	*0.00*
t_3	*price = 599 ∧ usageScenario = gaming*	*0.50*
t_4	*cpuCores = 4 ∧ usageScenario = multimedia*	*0.50*
t_5	*cpuCores = 4*	*~0.54*

Advantages: test case generation

- Let stakeholders evaluate the coverage

t_2	*price = 799 ∧ usageScenario = gaming*	*0.00*
t_3	*price = 599 ∧ usageScenario = gaming*	*0.50*

test case	*stakeholder*	*correct?*
t_2	s_0	*yes*
t_2	s_1	*yes*
t_2	s_2	*yes*
t_2	s_3	*yes*
t_2	s_4	*yes*
t_2	s_5	*no*
t_2	s_6	*yes*
t_2	s_7	*yes*

Advantages: test case generation

- Let stakeholders evaluate the coverage

t_2	*price = 799 ∧ usageScenario = gaming*	*0.00*
t_3	*price = 599 ∧ usageScenario = gaming*	*0.50*

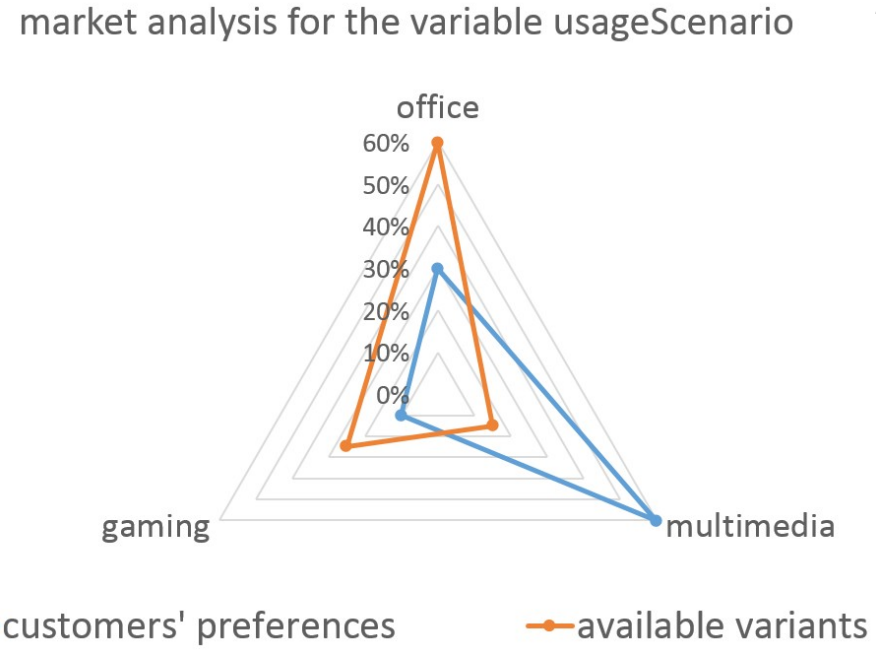
test case	*stakeholder*	*correct?*
t_2	s_0	*yes*
t_2	s_1	*yes*
t_2	s_2	*yes*
t_2	s_3	*yes*
t_2	s_4	*yes*
t_2	s_5	*no*
t_2	s_6	*yes*
t_2	s_7	*yes*

test case	*stakeholder*	*correct?*
t_3	s_0	*no*
t_3	s_1	*no*
t_3	s_2	*yes*
t_3	s_3	*yes*
t_3	s_4	*no*
t_3	s_5	*no*
t_3	s_6	*no*
t_3	s_7	*yes*

Advantages: market analysis

- Compare customers' preferences with the number of available variants

$$availability(v_i, d_1 \in dom(v_i)) = \frac{coverage(\{v_i = d_1\} \cup C)}{\sum_{d_j \in dom(v_i)} coverage(\{v_i = d_j\} \cup C)}$$



Advantages: session forecasting

- avoid the no-solution-could-be-found dilemma
  - Rank domain elements compared to the coverage
  - Approximate the number of consistent variants for each variable without an assignment

$$p(d_j \in dom(v_i)) = coverage(\{v_i = d_j\} \cup C) \times \prod_{v \in V} |dom(v)|$$

- rank diagnoses
  - Approximate the number of consistent variants, if one of the diagnoses is applied

$$p(d_j \in dom(v_i)) = coverage(C / \Delta) \times \prod_{v \in V} |dom(v)|$$

Conclusion

- Approximating the number of consistent variants compared to the number of all variants
- With Gibbs sampling you can ...
  - ... estimate the restriction rate and evaluate the complexity of a knowledge base
  - ... generate test cases for boundary value analysis
  - ... do market analysis
  - ... session forecasting (no-solution-could-be-found and rank diagnoses)

References

- [Chandola et al. 2009] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. ACM Comput. Surv., 41:15:1–15:58. 2009.
- [Felfernig et al. 2010] Alexander Felfernig, Christoph Zehentner, and Paul Blazek. Corediag: Eliminating redundancy in constraint sets. In Martin Sachenbacher, Oskar Dressler, and Michael Hofbaur, editors, DX 2011. 22nd International Workshop on Principles of Diagnosis, pages 219 – 224, Murnau, GER, 2010.
- [Junker 2004] Ulrich Junker. Quickxplain: preferred explanations and relaxations for over-constrained problems. In Proceedings of the 19th national conference on Artifical intelligence, AAAI'04, pages 167–172. AAAI Press, 2004.
- [Piette 2008] Cedric Piette. Let the solver deal with redundancy. In Proceedings of the 2008 20th IEEE International Conference on Tools with Artificial Intelligence - Volume 01, pages 67–73, Washington, DC, USA, 2008. IEEE Computer Society.
- [Schubert, Felfernig 2010] Monika Schubert and Alexander Felfernig. A Diagnosis Algorithm for Inconsistent Constraint Sets. In Proceedings of the 21st International Workshop on the Principles of Diagnosis. 2010.