(Hrsg.): SE'18: Software Engineering-Tagung der Gesellschaft für Informatik (GI),
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 11

Improving the Efficiency of Dislocality Constraints for an Automated Software Mapping in Safety-Critical Systems

Robert Hilbrich[1], Michael Behrisch[2]

Abstract: This is a brief overview of the paper, which should be 70 to 150 words long and include the most relevant points. This has to be a single paragraph.

Keywords: Schlagwort1; Schlagwort2

1 Introduction

Engineering complex and safety-critical systems is still challenging and costly. Despite our advancements in tools and method, its design still bears risk and uncertainties with regard to its outcome. The formalization and automation of crucial engineering tasks is a promising approach to tackle these challenges.

- What is mapping and deployment?
- Creates the prerequisites for execution
- influences reliability
- Example: DAL A software on DAL E hardware
- Main challenge: complexity of entire system

2 Automated Construction of Solutions

In order to achieve an automated construction of a mapping solution and to argue its correctness based on its synthesis, a formalization of the mapping problem and the synthesis steps for a solution are required. For smaller mapping problems, this has been successfully achieved based on Linear Integer Programming [Da06; Ku09], SMT-based solvers [VS13] or evolutionary algorithms[Wh11]. However, these approaches reach their limits when large-scale mapping problems with limited gradient information to guide a search process are

[1] German Aerospace Center (DLR), Rutherfordstr. 2, 12489 Berlin, Germany robert.hilbrich@dlr.de
[2] German Aerospace Center (DLR), Rutherfordstr. 2, 12489 Berlin, Germany michael.behrisch@dlr.de

considered. The authors instead chose to transform a mapping problem into a semantically equivalent *Constraint Satisfaction Problem (CSP)* [Ap03; De03] and solve this CSP with *Constraint Programming* techniques [PFL16; RBW06]. The advantages of using Constraint Programming in comparison to other techniques lie in the availability of powerful modeling elements, such as an ALLDIFFERENT constraint, and the ease with which custom search heuristics can be implemented.

2.1 Constraint Satisfaction Problems

Constraint Programming refers to a set of techniques in artificial intelligence and operations research. These techniques assist in finding solutions for problems based on variables, which are affected by constraints. Each constraint defines valid or invalid solutions for a subset of these variables. In this paper, a subclass of constraint satisfaction problems is used to express mapping problems: *finite domain integer constraint satisfaction problems* in which each variable has a finite integer domain. Solutions for this problem class can be obtained by applying a combination of *search* techniques – including backtracking – and constraint *propagation* techniques for value elimination.

To illustrate the modeling approach of Constraint Satisfaction Problems, consider the well-known *Map Coloring* problem as an example. This problem asks, whether it is possible to color a map with only four colors in such a way, that neighboring countries have different colors. It can be formulated as a CSP by assigning an integer variable x_i for each country with the index i. The domain of each variable corresponds to the four colors: $D_{x_i} = \{0, 1, 2, 3\}$. In order to model the restrictions of this problem, a constraint is added for each pair of adjacent countries. If country x_i is adjacent to country x_j, then $x_i \neq x_j$ is required. The search algorithm is now responsible to select a variable and test a value of its domain. Assuming a simple "first variable, first value" strategy, the variable x_0 would be chosen and set to the value 0 as a test. This would be *propagated* to all variables which are directly linked to x_0 by a constraint, so that the value 0 gets removed from their domains. This removal may lead to other value removals in indirectly linked variables and is processed until a fix point is reached. If a contradiction is encountered or the domain of a variable becomes empty, *backtracking* is initiated, so that the next value of the variable x_0 is tested. Otherwise, the search algorithm continues with the next uninstantiated variable. This example also shows, that the propagation of the NOTEQUAL constraint is *weak*, because it affects only two variables and invalidates only 4 out of the 16 possible value combinations between two variables.

The formulation of mapping problem as a Constraint Satisfaction Problem is not enough. Solutions for the variables in the CSP model need to be "interpreted" in the problem space. In the example above, the user needs to know, which country corresponds to x_i and which color corresponds to the value in its domain. Therefore, as a last step, an automatic transformation of the solution from the CSP model to the problem space is required. The entire process to automatically obtain a mapping solution is depicted in Figure 1.

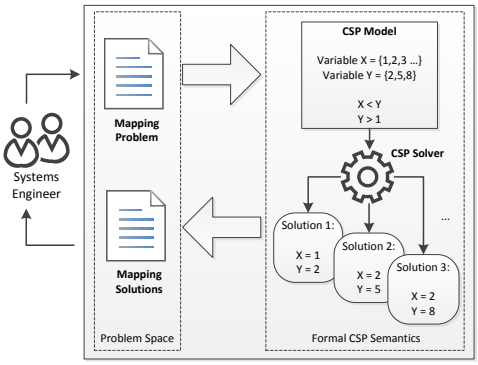

Fig. 1: Process to automatically construct a solution for a mapping problem

2.2 Toolsuite ASSIST

As a proof of concept, the toolsuite *Architecture Synthesis for Safety-Critical Systems (ASSIST)* [Hi14] was developed by the authors. It is open source and uses the constraint solver *Choco* [PFL16] internally. ASSIST (see Figure 2) allows a systems engineer to automatically construct and optimize mappings based on textual specifications of the

- mappable elements,
- (safety) constraints for valid/invalid assignments and
- optimization goals.

The textual specifications in ASSIST conform to a domain-specific language which was jointly developed with the partners in the projects. This approach allows to hide the intricacies of a formal specification. Using a domain-specific language is expedient to enable systems engineers without a formal education in computer science to specify a mapping problem.

3 Ensuring Redundancy by Requiring Dislocality

- In order to ensure a reliability / redundancy work properly - it is essential to ensure is differences
- Differences in the deployment of redundant software components to make sure that the errors do not appear in the same manner at the same location (zonal effects)
- Dislocality is a typical constraint in ASSIST to express these differences
- Simple Modeling approach for a CSP (location variables)
- Use allDiffererent constraint - straight forward

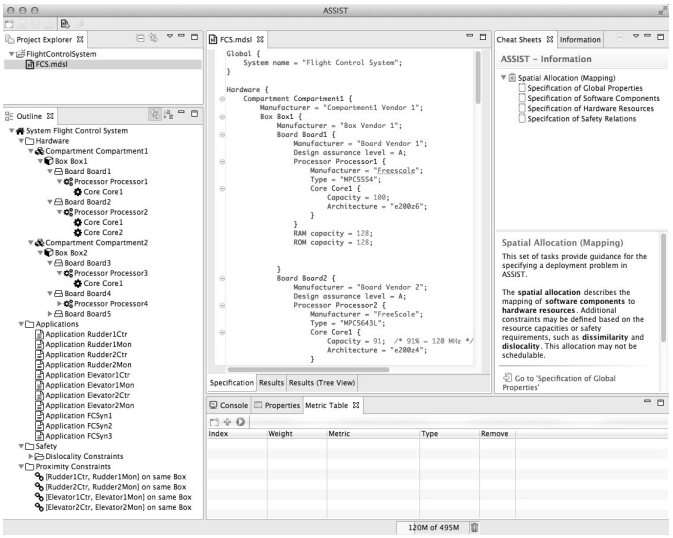

Fig. 2: Screenshot of ASSIST with a specification for a control system

- Reality: AllDifferent for List of Lists
- Example to describe the differences

4 Modeling Advanced Dislocality Constraints

- Based on available tools - one option is the recursive approach
- Implementation is straight forward, but it requires a lot of constraints - slow in practice
- Another option is to react on instantiation only and remove unwanted solutions - weak in propagation
- Last Option is to combine inst-only approach with int-values union and alldifferent

5 Experiments

- Synthetic example generator
- 10/20 Examples were generated with a parameter domain ...
- Search with the same strategies DomWD, minValueFirst
- iMac 5k, 64 GB RAM, Choco 4.0.6

6 Results

- Show results for var count, constraint count
- Show results for resolution time
- Show results for backtracks and fails

7 Conclusions

I am a summary

References

[Ap03] Apt, K. R.: Principles of constraint programming. Cambridge University Press, 2003.

[Da06] Damm, W.; Metzner, A.; Eisenbrand, F.; Shmonin, G.; Wilhelm, R.; Winkel, S.: Mapping Task-Graphs on Distributed ECU Networks: Efficient Algorithms for Feasibility and Optimality. In: Embedded and Real-Time Computing Systems and Applications, 2006. Proceedings. 12th IEEE International Conference on. Pp. 87–90, 2006.

[De03] Dechter, R.: Constraint Processing. Elsevier Science & Technology, 2003.

[Hi14] Hilbrich, R.: Architecture Synthesis for Safety Critical Systems - ASSIST, online, 2014, URL: http://assist.hilbri.ch.

[Ku09] Kugele, S.; Haberl, W.; Tautschnig, M.; Wechs, M.: Optimizing Automatic Deployment Using Non-functional Requirement Annotations. In (Margaria, T.; Steffen, B., eds.): Leveraging Applications of Formal Methods, Verification and Validation. Vol. 17, Communications in Computer and Information Science, Springer Berlin Heidelberg, pp. 400–414, 2009, ISBN: 978-3-540-88478-1, URL: http://dx.doi.org/10.1007/978-3-540-88479-8_28.

[PFL16] Prud'homme, C.; Fages, J.-G.; Lorca, X.: Choco Documentation, TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., 2016, URL: http://www.choco-solver.org.

[RBW06] Rossi, F.; van Beek, P.; Walsh, T., eds.: Handbook of Constraint Programming. ELSEVIER SCIENCE & TECHNOLOGY, 2006.

[VS13] Voss, S.; Schatz, B.: Deployment and Scheduling Synthesis for Mixed-Critical Shared-Memory Applications. In: Engineering of Computer Based Systems (ECBS), 2013 20th IEEE International Conference and Workshops on the. Pp. 100–109, 2013.

[Wh11] White, J.; Dougherty, B.; Thompson, C.; Schmidt, D. C.: ScatterD: Spatial deployment optimization with hybrid heuristic/evolutionary algorithms. TAAS 6/3, p. 18, 2011.