Machine-Assisted Reformulation for MiniZinc

Huu-Phuc Vo

October 21, 2019

Machine-Assisted Reformulation for MiniZinc

Huu-Phuc Vo
Dept. of Information Technology
Uppsala University
Uppsala, Sweden
Email: huu-phuc.vo@it.uu.se

***Abstract*—Model reformulation plays an important role in improving models and reducing search space so that solutions can be found faster. In solving Constraint Satisfaction Problems (CSPs), a model of a CSP may be solved rapidly, while a different model may take excessively long to solve. The efficient solution of CSP is significant in real-world applications, such as air traffic management, resource allocation, production scheduling, and bioinformatics. Many technologies such as constraint programming (CP), hybrid technologies, mixed integer programming (MIP), constraint-based local search (CBLS), boolean satisfiability (SAT) could have different solvers and backends to solve the real-time problems. Model reformulation can have a significant impact on solving time. Techniques from formal methods will be used to provide machine assistance for *MiniZinc*, which is the high-level modelling language to model CSPs. The verification tool, Isabelle, will be used to verify the correctness of reformulations. We plan to apply recent results in formal methods such as program analysis and synthesis to provide semi-automated frameworks for model analysis. In this paper, we identify the challenges, implement frameworks, and evaluate our experimental results in reformulations for future research.**

***Index Terms*—formal methods, implied constraint, reformulations, constraint programming, optimisation, modelling languages**

I. Motivation

Model reformulation [1] plays an important role in improving models and reducing search space so that solutions can be found faster. In solving Constraint Satisfaction Problems (CSPs) [2], a model of a CSP may be solved rapidly, while a different model may take excessively long to solve. The efficient solution of CSP is significant in real-world applications, such as air traffic management, resource allocation [3], production scheduling, and bioinformatics [4]. Because such problems are often NP-hard, all known algorithms have exponential-time worst-case behaviour. Many technologies such as CP, MIP [5], CBLS, satisfiability modulo theories (SMT) [6], [7], and SAT [6] could have different solvers and backends to solve the real-time problems. Each technology has its scope of application, and none of the technology is dominant all problems. CP [8] is pervasive and widely used to solve real-time problems which input data could be scaled up to the enormous sizes, and results are required to be given efficiently and dynamically.

CSPs can be reformulated automatically or semi-automatically. The automatic reformulation of CSPs can be obtained using various methodologies and approaches [9]–[14]. The automation in modelling has been advanced in five areas such as (1) generating a kernel, in which the compilation in a direct or non-direct system, (2) identifying symmetries, (3) breaking symmetries, (4) adding implied constraints, and (5) transforming constraints. These five advances are discussed in an invited talk at the 10th international workshop on constraint modelling and reformulation [15]. In addition, the reformulation may be achieved semi-automatically [16]–[19]. One approach is using a learning-based reformulation approach [19] or improving and stepping towards the automatic methods work [20].

The reformulation of a model into a logically equivalent model is a valuable tool to assist a modeller. A correct reformulation can significantly improve the solving time with the confidence of modeller. The Global Constraint Catalogue [21], the global-constraint library of the *MiniZinc* system [22], and the Essence system [23], [24] are a few resources for reformulation. Both *MiniZinc* [22] and Essence [23], [24] are solver-independent languages for modelling CSPs with many common global constraints. In addition, their reformulations are used when a target solver does not have the required globals. Reformulations are inferred in both the Model Seeker [25] and the Globalizer [26] by testing many combinations of global constraints on possible input data and then ranking the possible reformulations.

No proofs of correctness are provided for these reformulations, and the modeller must decide the correctness of the suggested reformulations. No work has been done on verifying the correctness of such reformulation rules, although there are tools [22]–[24] that could generate reformulation rules.

II. Objectives

Interactive theorem provers are notoriously difficult to use. Moreover, the development of proof tactics can be difficult in a new application area such as CP. However, even suggesting and verifying only a few reformulation rules will remarkably benefit modellers of CP and will still be a major advance over state of the art. Machine assistance that provides verified reformulation rules allows modellers to improve their models by reformulations with confidence. The contributions of the Ph.D. project are as follows

- Develop general-purpose tactics for reformulations of models in CP.
- Propose proof systems to prove soundness and completeness of reformulation rules.

- Provide semi-automated and automated frameworks for model analysis, synthesis, and refinement.
- Implement a library of automatic generation of verified reformulation rules for the *MiniZinc* toolchain.

The key objectives of the Ph.D. project are to support modellers to generate equivalent models that are verified and solve CSPs faster. The novelty and originality of the project are that although many constraint modelling toolchains support to generate reformulation rules [22]–[24], there is not any work that has been done on verifying the correctness of such generated reformulation rules. For instance, with possible input date, the Model Seeker [25] and the Globalizer [26] test many combinations of global constraints. Then, all possible reformulations are ranked and finally provided to modellers without verifying their soundness and completeness. Applying theorem proving techniques and verifying reformulation rules crucially differentiate our work from other existing works. Even though just a few generated reformulation rules is verified, it could be a significant advance to non-expert modellers over state of the art.

When writing this paper, it does not exist any model reformulation system with the auto-generated proof of models equivalence. The problem consists of several components and research questions that are needed to solve gradually and incrementally. It requires comprehensive knowledge, which is related to programming languages, theorem proving techniques, optimisation methods, and machine learning. In order to solve the problem, we divide the problems into sub-problems with corresponding questions. For instances, (1) how to find the implied constraints in a given model? (2) how to assert that the improved model with supplemental implied constraints is better than the original model? (3) how to automatically derive the formulas of basic and improved models? (4) what and how the translation between modelling language and proof language? (5) how to develop an integrated system that takes the original model, produces an improved model with the proof of equivalence between these models?

To the engineering aspects, the system should contain four components as follows

- Preprocessing component: to transform the model to corresponding theorem prover input and vice versa.
- Reformulation component: to produce an improved model from a given model.
- Theorem prover component: to prove the soundness and completeness of two models.
- Intermediate language component: to translate between modelling language and theorem proving representation.

III. Methodology

In the scope of the project, we aim to answer all questions as mentioned above. In the initial phase, we firstly focus on solving the problem manually. In the second phase, we selectively improve some manual components to semi-automatic components. In the third phase, we tackle the automated-components problem. Finally, we accomplish the project by integrating and benchmarking the system.

In the first phase, MiniZinc's reformulation rules will be formalised in Isabelle [**?**] or a similar system. We take advantage of Isabelle theorem prover to prove the soundness and completeness of two models. We aim to develop general-purpose tactics for the reformulation of models.

We investigate more specialised theories that will be useful in proving the correctness and soundness in the second phase. We will investigate logics for proving the correctness and soundness of reformulation rules and their integration into MiniZinc.

Finally, we will benchmark the system using several case studies that could be found from MiniZinc competition [27], and CSPLib [28].

Our preliminary results are summarised as follows

- a survey of model reformulations that compares, contrasts, and systematically categorises several approaches
- a publication which aims to solve a typical CSP using different solvers such as Gecode, Chuffed, Gurobi, OscaR.cbls, and Lingeling over two versions of MiniZinc toolchain.

IV. Research Plan

We plan to achieve research goals in Table I. We fully accomplished Problem relevance and a part of Research relevance such as writing a survey at the moment. Furthermore, we are partially working on the first Research contribution, which is to develop general-purpose tactics for reformulations of models in constraint programming.

Activity	**2018**	**2019**	**2020**	**2021**
Problem Relevance: Identification and analysis of challenges in reformulations, program synthesis, and refinement. Investigate and study formal logic, theorem proving techniques, and proof assistants	✓			
Research Contribution: Develop general-purpose tactics for reformulations of models in constraint programming. Write the survey of the fields.	✓	✓		
Research Contribution: Develop proof systems to prove soundness and completeness of reformulation rules.		✓	✓	
Research Contribution: Develop semi-automated and/or automated frameworks for model analysis, synthesis, and refinement		✓	✓	
Research Contribution: Develop a common foundation for automatic generation of verified reformulation rules, and integrate to the *MiniZinc*toolchain. Ph.D. thesis composition			✓	✓
Research Communication: Benchmarking: design and applications. The preliminary and final Ph.D. defense				✓

Table I: Timeline for Ph.D. Research

References

[1] Barbara M Smith. Modelling. In Francesca Rossi, Peter van Beek, and Toby Walsh, editors, *Handbook of Constraint Programming*, chapter 11, pages 375–404. Elsevier, 2006.

[2] Eugene C. Freuder and Alan K. Mackworth. Constraint satisfaction: An emerging paradigm. In Francesca Rossi, Peter van Beek, and Toby Walsh, editors, *Handbook of Constraint Programming*, chapter 2, pages 11–26. Elsevier, 2006.

[3] Holmes E Miller, William P Pierskalla, and Gustave J Rath. Nurse scheduling using mathematical programming. *Operations Research*, 24(5):857–870, 1976.

[4] Francisco Azevedo and Pedro Barahona. Applications of an extended set constraint solver. In *ERCIM / CompulogNet Workshop on Constraints*, 2000.

[5] John N. Hooker. *Integrated Methods for Optimization*. Springer, second edition, 2011.

[6] Clark Barrett. " decision procedures: An algorithmic point of view," by daniel kroening and ofer strichman, springer-verlag, 2008. *J. Autom. Reasoning*, 51(4):453–456, 2013.

[7] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving sat and sat modulo theories: from an abstract davis–putnam–logemann–loveland procedure to dpll (t). *Journal of the ACM (JACM)*, 53(6):937–977, 2006.

[8] Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006.

[9] Marco Cadoli and Toni Mancini. Automated reformulation of specifications by safe delay of constraints. *Artificial Intelligence*, 170(8):779–801, 2006.

[10] Christian Bessiere, Joël Quinqueton, and Gilles Raymond. Mining historical data to build constraint viewpoints. In *5th International Workshop on Constraint Modelling and Reformulation*, pages 1–16, 2006.

[11] Bernadette Martınez-Hernández and Alan M Frisch. The automatic generation of redundant representations and channelling constraints. In *Proceedings of the 5th International Workshop on Constraint Modelling and Reformulation*, pages 42–56, 2006.

[12] Christopher Mears and Maria Garcia De La Banda. Towards automatic dominance breaking for constraint optimization problems. In *IJCAI*, pages 360–366, 2015.

[13] Christopher Mears and Maria Garcia De La Banda. Towards automatic dominance detection in constraint optimisation problems. In *ModRef15*, 2015.

[14] John Charnley, Simon Colton, and Ian Miguel. Automatic generation of implied constraints. In *ECAI*, volume 141, pages 73–77, 2006.

[15] Alan M Frisch. A decade of progress in constraint modelling and reformulation the quest for abstraction and automation, 2011. Available at https://www-users.cs.york.ac.uk/~frisch/Research/decade.pdf.

[16] Remi Coletta, Christian Bessiere, Barry O'Sullivan, Eugene C Freuder, Sarah O'Connell, and Joel Quinqueton. Semi-automatic modeling by constraint acquisition. In *International Conference on Principles and Practice of Constraint Programming*, pages 812–816. Springer, 2003.

[17] Remi Coletta, Christian Bessiere, Barry O'Sullivan, Eugene C Freuder, Sarah O'Connell, and Joel Quinqueton. Constraint acquisition as semi-automatic modeling. In *Research and Development in Intelligent Systems XX*, pages 111–124. Springer, 2004.

[18] Özgür Akgün. *Extensible automated constraint modelling via refinement of abstract problem specifications*. PhD thesis, University of St Andrews, 2014.

[19] Kiana Zeighami, Kevin Leo, Guido Tack, and Maria Garcia de la Banda. Towards semi-automatic learning-based model transformation. In *International Conference on Principles and Practice of Constraint Programming*, pages 403–419. Springer, 2018.

[20] Maxim Shishmarev, Christopher Mears, Guido Tack, and Maria Garcia de la Banda. Learning from learning solvers. In Michel Rueher, editor, *Principles and Practice of Constraint Programming*, pages 455–472, Cham, 2016. Springer International Publishing.

[21] Nicolas Beldiceanu, Mats Carlsson, Sophie Demassey, and Thierry Petit. Global constraint catalogue: Past, present and future. *Constraints*, 12(1):21–62, Mar 2007.

[22] Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. Minizinc: Towards a standard cp modelling language. In Christian Bessière, editor, *Principles and Practice of Constraint Programming – CP 2007*, pages 529–543, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[23] Alan M. Frisch, Matthew Grum, Chris Jefferson, Bernadette Martinez Hernandez, and Ian Miguel. The design of Essence: A constraint language for specifying combinatorial problems. In *IJCAI 2007*, pages 80–87. Morgan Kaufmann, 2007.

[24] Peter Nightingale, Özgür Akgün, Ian P. Gent, Christopher Jefferson, and Ian Miguel. Automatically improving constraint models in Savile Row through associative-commutative common subexpression elimination. In Barry O'Sullivan, editor, *CP 2014*, volume 8656 of *LNCS*, pages 590–605. Springer, 2014.

[25] Nicolas Beldiceanu and Helmut Simonis. A model seeker: Extracting global constraint models from positive examples. In *International Conference on Principles and Practice of Constraint Programming*, pages 141–157. Springer, 2012.

[26] Kevin Leo, Christopher Mears, Guido Tack, and Maria Garcia De La Banda. Globalizing constraint models. In *International Conference on Principles and Practice of Constraint Programming*, pages 432–447. Springer, 2013.

[27] Peter J Stuckey, Thibaut Feydy, Andreas Schutt, Guido Tack, and Julien Fischer. Minizinc challenge. https://www.minizinc.org/challenge.html.

[28] Barbara M. Smith. All combinatorial satisfaction problems. http://www.csplib.org/.