Game-based Configuration Task Learning with ConGuess: An Initial Empirical Analysis

Andreas Hofbauer[1,*], Alexander Felfernig[1]

[1]Graz University of Technology, Inffeldgasse 16b, Graz, 8010, Austria

Abstract

The concepts and semantics of constraint solving and configuration need to be understood in order to be able to develop one's own configuration knowledge bases. Developing a related basic understanding is in many cases quite challenging. Consequently, further support is needed that makes the learning of configuration knowledge representation practices and semantics less effortful. In this paper, we provide a short overview of ConGuess which is a game-based learning environment for constraint-based configuration tasks. In this context, we report the results of a user study which focused on an analysis of the perceived complexity of different constraint types and on a corresponding usability analysis.

Keywords

Knowledge-based Configuration, Constraint Solving, E-Learning, Gamification

1. Introduction

Assuring the correct understanding of configuration knowledge representations and corresponding semantics is an important issue specifically in industrial configuration settings. Such an understanding can be regarded as a precondition for successful configurator development and maintenance [1, 2, 3]. Following the basic idea of gamification-based learning [4], we have developed ConGuess [5] which is an application supporting the learning of the semantics of constraint satisfaction problems (CSP) [6] in a gamification-based fashion.

The overall idea of ConGuess is to pre-generate configuration tasks (represented as CSPs) and let users (game players) try to figure out correct solutions for the defined tasks. With this, ConGuess follows the idea of earlier related work focusing on the learning of graphical configuration constraints (specifically, incompatibility constraints) and the concepts of hitting sets in model-based diagnosis (specifically, minimal food item sets that cover all relevant vitamins) [7, 8, 9].

Also in this line of research, Jefferson et al. [10] present the application Combination which supports the learning of configuring color ray emitting wooden pieces such that no color array hits a wooden piece of different color. Compared to related work, ConGuess extends the expressivity of constraint representations and also includes a gamification-based approach that can help to increase user engagement.

The contributions of this paper are the following: (1) we provide a short introduction to the ConGuess gaming app, (2) we report the results of an initial complexity and usability analysis that has been conducted in an Artificial Intelligence university course, and (3) we discuss different open issues for further related research.

The remainder of this paper is organized as follows. In Section 2, we provide a short overview of the ConGuess app specifically introducing the major idea behind. Thereafter, in Section 3, we discuss first insights regarding the perceived complexity of different constraint types. In Section 4, we report results regarding the usability of ConGuess. In Section 5, we discuss potential threats to validity. The paper is concluded with a discussion of open research issues in Section 6.

2. The ConGuess Game

In the line of related research (e.g., [7]), ConGuess is provided as Android app[1] which includes mechanisms for automated CSP generation and evaluation of solutions.[2] In ConGuess, players have to solve pre-generated CSPs [6] which are represented in terms of a set of Variables V with related domain definitions and a corresponding set of constraints (C). The task of players is to identify solutions (configurations) that satisfy all given constraints.

ConGuess supports different game levels where with an increasing level the corresponding CSPs become more difficult to solve. For solving configuration tasks (CSPs), players (users) have a pre-defined time limit. For each correctly solved CSP, players receive corresponding points

[1]See play.google.com.

[2]We apply the constraint solving library Choco (choco-solver.org).

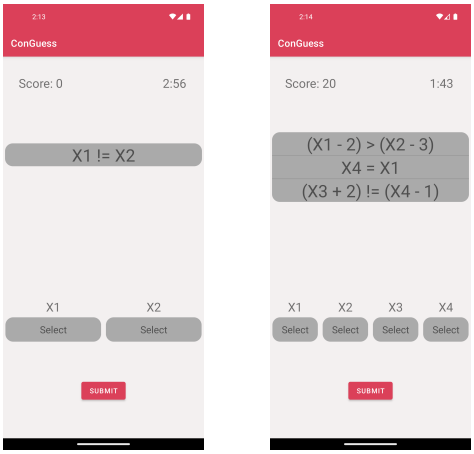
ConfWS'23: 25th International Workshop on Configuration, Sep 6–7, 2023, Málaga, Spain

*Corresponding author.

andreas.hofbauer@student.tugraz.at (A. Hofbauer); alexander.felfernig@ist.tugraz.at (A. Felfernig)

https://felfernig.ist.tugraz.at/ (A. Felfernig)

0000-0003-2956-3862 (A. Hofbauer); 0000-0003-0108-3146 (A. Felfernig)



(a) A simple configuration task (CSP).

(b) A more complex configuration task.

Figure 1: ConGuess: configuration task user interface.

which increases their overall game score. If a player proposes a configuration *inconsistent* with the given set of constraints, his/her score is not reduced and further tries are possible. With an increasing number of unsuccessful tries, the number of points that can be received for a correct solution gets decreased. Finally, the game provides a global highscore ranking which helps to further motivate users to improve their personal highscore.

A screenshot of ConGuess in action is provided in Figure 1 which depicts two different configuration tasks (a more simple one on the left hand side and a more complex one on the right hand side). The value of each corresponding variable has to be specified individually indicated by the *select* button. A major objective of the app is to make the configuration task representation as understandable as possible. For this reason, the overall rule of the app in terms of information visualization is that each configuration task fits into the screen without the need of scrolling.

As mentioned, constraint satisfaction problems (CSPs) in ConGuess are pre-generated. The consistency of individual configuration tasks (CSPs) is checked with the Choco constraint solver. If a generated configuration task (variables and corresponding constraints) is consistent, the corresponding setting is stored for further usage (as configuration task given to players). Player-proposed solutions as well as generated configuration tasks are checked for consistency using Choco.[3]

[3]Details on the ConGuess constraint solving and configuration task generation approach can be found in Hofbauer and Felfernig [5].

3. Complexity of Constraint Types

Our goal was to better understand in which way different types of constraints are understood by players. In order to achieve this goal, we performed a user study with 150 bachelor students engaged in an Artificial Intelligence course at the Graz University of Technology. Best-performing students had the chance to achieve additional bonus points considered then as a part of the overall evaluation. In total, 780 game sessions have been completed within the scope of the user study resulting in an average number of 5.2 gaming sessions per study participant (with an average of 6 levels per session).

In each ConGuess session, correct and wrong guesses were tracked in combination with the corresponding configuration task shown to the player. The error rate R_{error} of specific configuration tasks (CSPs) was tracked following the metric shown in Formula 1. In this context, n_{total} is the total amount of guesses and n_{error} is the amount of wrong guesses for a CSP.

$$R_{error} = \frac{n_{error}}{n_{total}} \tag{1}$$

Within the scope of our study, we compared different *configuration task types* with regard to their understandability: configuration tasks (1) consisting of *equality and inequality* constraints, (2) consisting of constraints including a *range restriction*, i.e., $<, >, \leq, \geq$, (3) with *implications* (*requires*) and *equivalences*, and (4) with different *numbers of constraints and variables*.

In a first step, we focused on the analysis of single-constraint configuration tasks, i.e., configuration tasks with only one constraint included ($|C| = 1$). Tables 1–5 include example constraints which represent a cor-

responding analysis class, for example, the constraint $X1 = X2$ in Table 1 represents a configuration task with a singleton constraint of type equality constraint. Similarly, the first constraint in Table 2 represents a configuration task with a single constraint of type $<$.

Equality and Inequality Constraints. First, we have analyzed player failure rates when being confronted with singleton equality and inequality constraints. The corresponding R_{error} rates are depicted in Table 1. As can be immediately seen, the error rates for such constraints are rather low (on an average, below 5%) indicating a high degree of understandability in the reported basic setting.

example constraint(s)	avg. R_{error} in %
X1 = X2	4.08
X3 != X2	2.11

Table 1
Error rates of binary (in-)equality constraints.

Range Restriction Constraints. In the next step, we analyzed the understandability of range restriction constraints (<, >, $\leq$, $\geq$) (see Table 2). Compared to settings including the $<$ and $>$ operators, error rates significantly increase with settings including $\leq$ and $\geq$ operators. One way to explain this significant difference is the increased complexity of $\{\leq, \geq\}$ due to the fact that both dimensions, inequality and equality have to be taken into account at the same time.

example constraint(s)	avg. R_{error} in %
X3 < X4	8.41
X1 > X2	4.90
(X2 + 1) > X3	7.37
X3 $\leq$ X2	20.49
X2 $\geq$ X1	13.50

Table 2
Error rates of binary range restriction constraints.

Requires and Equivalence Constraints. In this context, we have compared the understandability of individual implications (requires) and equivalences (see Table 3). We can see that equivalence constraints have slightly lower error-rates than requires constraints. This is a result that has also been confirmed by a previous study of Felfernig et al [3]. One way to explain this difference is a potentially higher overhead induced by the analysis of implications since equivalences can be reduced to settings where both sides of the logical operator must have the same logical value. Further related work on model understandability can be found in Sepasi et al. [11] where cognitive complexity is also measured on the basis of eye tracking technologies.

example constraint(s)	avg. R_{error} in %
(X2 = X4) $\leftrightarrow$ (X1 < X3)	8.05
(X1 > X3) $\leftrightarrow$ (X1 < X2)	17.39
(X3 != X2) $\rightarrow$ (X4 != X2)	18.06
(X1 > X3) $\rightarrow$ (X1 < X2)	20.53

Table 3
Error rates of requires and equivalence constraints.

Number of Constraints. When using $\{\rightarrow, \leftarrow\}$ instead of $\leftrightarrow$ for expressing equivalence knowledge, we need twice the amount of constraints. As could be observed in our analysis, an increasing number of constraints leads to increasing error rates due to a lower understandability of the configuration task (see Table 4).

example constraint(s)	avg. R_{error} in %
c1: X1 > X2 c2: X2 = X3	4.90
c1: X1 $\geq$ X2 c2: X1 > X4 c2: X3 > X2	17.12
c1: X1 != X3 c2: X2 > X3 c3: X4 $\leq$ X1 c4: X3 < X4	36.87

Table 4
Error rates with an increasing number of constraints.

N-ary Constraints. The highest error-rates in our study were encountered with constraints involving more than 2 variables. As we can see in Table 5, even configuration tasks with $|C| = 1$ are already difficult to solve for players, if the number of included variables is greater than 2. Furthermore, by increasing the number of constraints in a configuration task, it becomes extremely challenging for players to find a consistent solution.

example constraint(s)	avg. R_{error} in %
c1: X3 $\leq$ (X1 + 3) = (X2 - 2)	20.13
c1: X3 $\leq$ (X4 + 4) c2: X4 < X1 $\leq$ X3	72.12
c1: X3 $\geq$ X1 $\leq$ X2 c2: (X1 $\cdot$ 2) > X2 c3: X3 < X2	73.02
c1: X4 != (X3 + 3) < X1 c2: (X4 - 2) $\geq$ X1 c3: X1 < (X3 $\cdot$ 3) c4: X4 != (X1 + 3)	84.68

Table 5
Error rates with n-ary constraints.

4. Usability of ConGuess

To evaluate the usability of ConGuess, we have performed a usability study with 10 participants (computer science students on the bachelor level). For this purpose, we have used the System Usability Scale (SUS) [12]. SUS is a widely used tool for evaluating the usability of systems and software applications and it consists of a questionnaire with 10 items. After using ConGuess (without further explanations), the participants had to rate their perception of the software on a 5-point *Likert scale*. Following the SUS calculation scheme resulted in an overall evaluation of 89.5 out of 100 which is an excellent SUS rating expressing clear understandability and high willingness to use the system.

5. Threats to Validity

The results reported in this paper are based on the app usage by bachelor-level students within an Artificial Intelligence course. On the one hand, different educational backgrounds could be expected by persons working in industrial configurator projects. On the other hand, persons in an early phase of their career could be in a similar situation as students engaged in our study.

The settings analyzed in our study are limited in the sense that further aspects such as constraint grouping, i.e., constraint ordering, have not been analyzed in detail. Furthermore, we did not compare alternative ways of increasing the complexity level of individual configuration tasks which is important since our goal is to create a kind a flow where game players try to solve even more configuration tasks. We regard these aspects as major issues for future work.

6. Conclusion and Future Work

In this paper, we have presented ConGuess which is an Android app supporting the learning of configuration knowledge representations and underlying semantics. The basic idea of ConGuess is that players learn constraint semantics on the basis of trying to solve CSPs. We conducted an empirical study to better understand the cognitive complexity of different constraint structures.

Major issues for future work are the following: (1) we will analyze to which extent the grouping of constraints and corresponding variables can have an impact on the overall understandability of a configuration task, (2) based on the insights of our empirical study, we will propose adaption operations on real-world configuration knowledge bases and analyze related impacts on understandability (again, within the scope of empirical studies), (3) based on the results of further empirical studies, we will try to adapt the complexity measures currently integrated into the ConGuess configuration task generation. Improved complexity measures could result in a more user-centered increase of configuration task complexity and with this potentially also to a corresponding improved learning experience.

References

[1] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based Configuration - From Research to Business Cases, Elsevier, 2014.

[2] A. Felfernig, S. Reiterer, M. Stettinger, J. Tiihonen, Towards understanding cognitive aspects of configuration knowledge formalization, in: Vamos'2015, ACM, New York, NY, USA, 2015, p. 117–123.

[3] A. Felfernig, M. Mandl, A. Pum, M. Schubert, Empirical knowledge engineering: Cognitive aspects in the development of constraint-based recommenders, in: Trends in Applied Intelligent Systems, volume 6096 of *LNCS*, Springer, Berlin / Heidelberg, 2010, pp. 631–640.

[4] R. Raymer, Gamification: Using game mechanics to enhance elearning, ELearn 2011 (2011).

[5] A. Hofbauer, A. Felfernig, ConGuess: A Learning Environment for Configuration Tasks, in: ACM SPLC'22, Association for Computing Machinery, New York, NY, USA, 2022, p. 156–157.

[6] F. Rossi, P. van Beek, T. Walsh, Handbook of Constraint Programming, Elsevier, Amsterdam, The Netherlands, 2006.

[7] A. Felfernig, M. Jeran, T. Ruprechter, A. Ziller, S. Reiterer, M. Stettinger, Learning games for configuration and diagnosis tasks, in: 17th International Configuration Workshop, volume 1453, CEUR, Vienna, Austria, 2015, pp. 111–114.

[8] A. Felfernig, M. Schubert, C. Zehentner, An efficient diagnosis algorithm for inconsistent constraint sets, AIEDAM 26 (2012) 53–62.

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

[10] C. Jefferson, W. Moncur, K. Petrie, Combination: Automated generation of puzzles with constraints, in: ACM Symposium on Applied Computing, ACM, New York, NY, USA, 2011, pp. 907–912.

[11] E. Sepasi, K. Balouchi, J. Mercier, R. Lopez-Herrejon, Towards a cognitive model of feature model comprehension: An exploratory study using eye-tracking, in: ACM SPLC'22, Association for Computing Machinery, New York, NY, USA, 2022, p. 21–31. URL: https://doi.org/10.1145/3546932.3546995. doi:10.1145/3546932.3546995.

[12] J. Brooke, SUS: A quick and dirty usability scale, Usability Eval. Ind. 189 (1995).