Conversational Modeling for Constraint Satisfaction

Eugene C. Freuder
Insight Centre for Data Analytics, University College Cork, Cork, Ireland
eugene.freuder@insight-centre.org

Abstract

Many problems, from Sudoku to factory scheduling, can be regarded as constraint satisfaction problems. A key component of real world problem solving is a *conversation* between a constraint programming expert and a problem domain expert to specify the problem to be solved. This paper argues that the time is ripe for progress in automating the constraint programmer side of this conversation and suggests promising avenues for this pursuit.

Introduction

Many problems, from Sudoku to factory scheduling, can be regarded as constraint satisfaction problems (CSPs). These involve finding values for problem variables that satisfy a set of constraints on allowable combinations of values. Constraint programmers model and solve such problems. Many methods, algorithms and languages have been developed for use by constraint programmers, but beyond developing tools for human constraint programmers, we can envision building automated constraint programmers.

Considerable progress has been made on aspects of this vision (viz. the Progress Towards the Holy Grail Workshops held at the International Conferences on Principles and Practice of Constraint Programming). However, relatively narrow progress has been made specifically on a key component of real world problem solving: a *conversation* between the constraint programming expert and the problem domain expert to specify (or *model*) the problem to be solved. More precisely, research on automating this conversation has primarily limited the domain expert's role to that of an "oracle" that only responds "yes" or "no" to questions posed.

A basic oracle question is simply a "membership query": is this a solution or not? Good progress has been made on expanding the range of questions that can be asked of such oracles, but the research emphasis has remained on placing as much of the burden as possible on the machine. This, of course, is a natural AI objective, and the intent here is not to discourage further progress in that direction. However, in practice domain experts may also be capable of playing a role in a richer and more balanced, and therefore more succinct and satisfying, *conversation*. Such an approach would also be in keeping with the increasing emphasis on *human-machine collaboration*.

This paper argues that the time is ripe for pursuing this view of the interaction between automated constraint programmer and human domain expert (or for that matter, automated domain expert), and suggests promising avenues for this pursuit. In particular, it argues for more research directed towards automating the ability to use, and draw out, input from the domain expert that goes beyond responding to "yes or no" queries, and which includes a more proactive role for the domain expert.

There are three subtracks of the AAAI Senior Member Presentation Track: bridge talks, summary talks, and blue sky idea talks. This paper identifies as one for a blue sky talk; but, of course, nothing is entirely new under the sun, and blue sky presentations are asked to "relate the talk as much as possible to the existing literature". Pointers to some of the relevant earlier work that might be built upon are provided here, especially work on expanding the range of questions that can be put to an oracle. In this respect the paper has some of the flavor of a summary presentation. In addition, the conversational goal advanced here is very much a "bridge" challenge. Most obviously it bridges constraint programming with natural language processing and machine learning, but other fields, such as recommender systems, knowledge representation, analogical reasoning, explanatory AI, human-in-the-loop AI, even computer vision, are also relevant.

While conversational problem acquisition is addressed here primarily in a high level, generic fashion, the implementation challenge can be simplified, at least initially, by targeting automation for specific, individual problem domains, or even to work with specific problem domain experts. Sleeman and Chalmers (2006) built a number of interfaces for specific problem domains that ask the user highly focused questions.

The conversational capabilities promoted here might be fully autonomous, or serve as an automated assistant for a human constraint programmer, again in line with the increasing emphasis on "human in the loop" AI.

In any event, the specific research directions suggested here are still far from fully addressing the requirements for practical constraint acquisition (Simonis 2023), and even further from fully automating the role of a human consultant on a large scale industrial project. However, advances in conversational modeling may play a larger role in making smaller scale solutions more accessible and affordable for individuals or non-profit organizations.

Volunteering

To start with, the domain expert should be allowed to *volunteer* information. This may seem obvious, but the oracle-based work has largely assumed that responding to oracle queries is the only role for the domain expert or "user". Bessiere et al. (2017) state that "our starting assumption is that the user is not able to articulate the constraints of the target network directly". This is certainly an important case to explore, both for theory (significant computational complexity results have been obtained) and practice (domain "experts" may actually be unable to explicitly articulate constraints). However, despite impressive progress on efficient acquisition, the number of oracular responses required may still be burdensome or even impractical for a human domain expert. And while an automated domain expert would have more patience, that automation also presents a challenge.

We are beginning to see more consideration of volunteered information. QUACQ2 (Bessiere et al. 2023) allows part of the problem to have already been supplied by the user and MGEQCA (Belaid et al. 2022) allows for "background knowledge", including known constraints.

Information can be volunteered at the *instantiation level,* as to whether specific assignments of values to variables are acceptable. However, it is efficient to converse at the *constraint level* about the presence or absence of entire constraints, or even sets of constraints. Volunteering can be as simple as providing an initial set of constraints. As the conversation progresses, more constraints can be provided, e.g. "that reminds me that I forgot to specify …".

Domain experts may want to volunteer information in other forms as well. For example, "I know this problem requires either constraint X or Y, but I'm not sure which." Another example: "The constraint between A and B has property C".

Domain experts can also *elaborate* on their oracular yes or no responses to constraint programmer questions. For example, they can volunteer *justifications* for their answers. Matchmaker (Freuder and Wallace 2002) accepted from the user a specific constraint that a rejected solution violates. Schekotihin and Friedrich (2009) provided *arguments* for rejecting proposals.

Soliciting

The constraint programmer can *solicit* or *prompt* information from the domain expert. Information gathered, volunteered or solicited, may be used to further guide the conversation.

The interaction with the constraint programmer may help domain experts make explicit constraints that they have been using implicitly and intuitively, or that they simply neglected to volunteer initially. Other information can be sought. For example, the constraint programmer might seek to elicit specific forms of elaboration for yes or no responses, e.g. "what could I change that would make this rejected instantiation acceptable?" or "how could this partial solution be extended to a full one?".

One reason that the time seems ripe for pursuing richer problem acquisition conversations is that there has indeed been impressive research on *asking* more sophisticated *questions* of the domain expert oracle that might be repurposed and extended for *prompting* more sophisticated and more direct *contributions* from the domain expert. (Conversely progress in the latter direction may motivate further progress in the former.)

For example, GENACQ (Bessiere et al. 2014) proposes *generalizations* of learned constraints. Broader or more open-ended prompting, rather than yes or no oracle questions, could allow the domain expert to bring to bear knowledge unavailable to the constraint programmer (e.g. "property A generalizes property B", "C is a kind of D", "E is an abstraction of F"). It can also allow the domain expert to directly provide a "maximal" generalization rather than responding yes or no to a series of under and over generalized proposals in an attempt to arrive at the maximal point.

Similarly, P-QUACQ (Daoudi et al. 2016) looks for incomplete *patterns* of constraints to propose missing ones. A suitable prompt could allow the domain expert to directly complete a pattern where the constraint programmer either does not see that specific option, or sees too many options to distinguish without requiring a burdensome number of oracle responses.

These advances have been taken to the next level through information acquired *during* the acquisition process. Daoudi et al. (2015) expanded generalization by detecting types of variables for generalization. Tsouros, Stergiou, and Bessiere (2019) exploited the structure of the learned problem. The suggestion here is that as the acquisition process proceeds the constraint programmer might also learn how to solicit such knowledge directly from the domain expert, or how better to utilize volunteered information.

SEQACQ (Prestwich 2020) is able to cope with *errors* in a training set, where instances are misclassified as solutions or non-solutions. The constraint programmer could use information acquired during the acquisition process to query the domain expert about errors or inconsistencies, e.g. "but in a similar situation earlier you didn't allow that".

The constraint programmer should be able to use, and indeed to solicit, domain expert contributions in many forms. We can envision developing a *catalogue* or taxonomy. These might be used to elicit useful information that the domain expert would not otherwise think to volunteer. "It would be helpful if you could tell me things like X."

One obvious avenue for identifying and studying conversational modes to encourage would be the observation of real-life interactions between consultants with various forms of problem solving experience and customers with various forms of problem domain expertise. However, there may also be classes of domain expert contribution that an automated constraint programmer is better able to make use of than a human constraint programmer.

A constraint programmer can utilize general knowledge ("those are both even numbers"), domain-specific knowledge ("all frammuses have the same warm-up time"), and user-specific knowledge ("you usually specify a maximum distance") to guide the solicitation and utilization of domain expert knowledge. It particular, such knowledge can motivate additional forms of generalization, identify additional patterns ("should this be fully symmetric?") or spot more errors ("that can't be right").

Such knowledge can also motivate prompts based on *analogy* or *classification*. For example, the domain expert might be prompted: "Others with a problem like yours have included a constraint of this type …" or "scheduling workers in this industry requires observing state regulation 18.75a". The literature on recommender systems should be a good resource for exploring prompting options.

Knowledge could be simply supplied to the constraint programmer, but it can also be learned through *experience*. This could be called *meta learning*, in contrast to the primary learning of the problem model. We would expect human consultant constraint programmers to utilize expertise gained through experience working in particular problem domains to become more effective and efficient at working with experts in those domains. A consultant might even learn to converse more effectively over time with a specific customer. We should imbue an automated constraint programmer with the ability to improve its conversational model acquisition abilities through experience.

Refining

The initial acquisition of a problem may be overconstrained or underconstrained, or otherwise imperfect or suboptimal. The problem may need maintenance as it changes over time. Further conversation between constraint programmer and domain expert can assist in addressing these issues.

There has been considerable work on automating *explanations* for failure in overconstrained problems (Gupta, Genc, and O'Sullivan 2021; Bleukx et al. 2023), which can assist the domain expert in modifying the problem to permit successful solution. The emphasis has understandably been on what explanation can be automatically provided *to* the user. However, we can also consider how helpful information might be received or solicited *from* the user.

For example, a model might admit some solutions, but the domain expert may recognize that it is still overconstrained because others are missing. The user may be able to partially specify a solution, e.g. "I know there is a solution with value v for variable X and value w for variable Y." This sort of interaction between constraint programmer and domain expert was considered in the context of *debugging* constraint problem specifications (Huard and Freuder 1993). In Ananke (Freuder, Wallace, and Nordlander 2009) if the user suggested partial (or complete) solutions that were not part of the current model, a *metamodel* was used to find changes in constraints that produce a new model with the proposed solutions.

Knowledge bases supporting constraint applications may also require *maintenance*. iCAM, Interactive Constraint Acquisition and Maintenance, (Nordlander, Freuder and Wallace 2007) interacted with hospital workers to maintain a constraint-based knowledge maintenance system for material management.

Domain experts may have *preferences* or *priorities* to communicate. Tsouros and Stergiou (2021) used "partial preference queries" to acquire preference information from users. Rossi and Sperduti (2004) allowed users to provide feedback and modify preferences after viewing proposed solutions. A constraint programmer may need to deal with multiple domain experts / customers on a single problem, and they might not agree.

Natural Language Processing

One reason for the relative paucity of progress in conversational constraint satisfaction is that this is a "bridge" problem that combines natural language processing with constraint programming. Natural language can, of course, be employed in a structured setting, e.g. menu choices, filling in templates. IBM's Modeling Assistant (Ceugniet 2019) enabled "formulation of models in natural language", for different decision domains, e.g. scheduling or planning. This can make it easier for someone without constraint programming expertise to build the model for an application. However, this is not the kind of natural language conversation that takes place where an automated consultant is building the model, based on a natural language conversation with a human customer.

The increasing availability of open source tools for natural language processing — and the rise of chatbots — makes it a propitious time to tackle this challenge. Restricting the conversations to specific problem domains, or perhaps even to interacting with specific individuals, can also focus and simplify the language challenge.

In fact, one may wonder: “Can’t chatbots just already fully address the challenge of conversational modeling?” Well “chatbot” is a moving target. Chatbots can certainly play a role. ChatGPT 3.5 provided some reasonable constraints in response to a question about what constraints to take into account in formulating a nurse rostering program for a hospital as a CSP. On the other hand, when it was prompted to generalize a constraint using the concepts of “odd” and “even”, these are excerpts from the resulting conversation:

Chatbot: In this example, rows 2 and 5 have red pebbles, as they have odd row numbers (2nd row and 5th row)

Author: I don't think you understand odd and even. 2 is not an odd number.

Chatbot: You are right, and I apologize for the confusion. I understand odd and even numbers correctly. I apologize for the incorrect statement in my previous response. … In this corrected example, rows 2 and 5 have red pebbles, as they have odd row numbers (2nd row and 5th row).

It is certainly worth investigating how chatbots can contribute to conversational constraint solving, and how limitations might be overcome. In particular, we can investigate how much further we can go with bespoke chatbots trained for the purpose. One obstacle such training would face is that the interactions between human consultants and customers that we seek to emulate are not stored somewhere on the internet to be scraped for chatbot training. They would normally take place face to face in the real world. We could envision recording them for chatbot edification. However, we should bear in mind the issues already being faced by chatbots as to who owns and benefits from the knowledge and expertise involved.

There has been interest in using natural language processing to acquire constraint problems specified in text form (Kiziltan, Lippi, and Torini 2016), which might be adapted to a conversational context. Bogaerts et al. (2020) acquire logic puzzles from text and Tsouros et al. (2023) bring Large Language Models to bear. It does not seem a large leap to expand from text to interactive conversation, and indeed the opportunity there to ask clarifying questions might in some sense make the language challenge easier.

Of course, language is not the only way for domain expert and constraint programmer to communicate. Vision can play a useful role. The early work on MAPSEE (Mackworth 1977) used constraint satisfaction to interpret sketch maps (albeit represented as plotter commands). Guns et al. (2023) worked with pictures of Sudoku puzzles.

Conclusion

A natural approach for AI is think about what information we need from the user, and how best to ask for it. The complementary approach is to think about what kinds of information the user can provide, and how best to solicit and utilize it. It is a subtle, but useful distinction. Both approaches are valuable, and can inform each other. This paper directs attention to the complementary approach and suggests directions in which to pursue it.

Our specific objective is to automate or assist constraint programmers in their conversations with domain experts to understand the problems to be solved. Much progress has been made viewing domain experts as oracles who can provide yes or no answers to queries from the constraint programmer. The goal here is a richer conversation, which can be both more efficient and more satisfying.

This conversation is less well-defined than one with an oracle, and the first challenge is formulating more specific research directions. Some have been suggested here, but at an abstract level with only “anecdotal” illustrations. In addition, somewhat ironically, it will be more difficult to evaluate progress here as it is easier to carry out experiments with an automated oracle than with an automated simulation of a conversational domain expert. Of course the participation of human domain experts, as well as professional constraint programming consultants, would be very desirable.

Chatbots may help, but we should devote resources to more direct symbolic reasoning research as well, employing the traditional approach of breaking down a daunting form of “intelligent behavior” into concrete, programmable pieces. In this way, and by integrating progress in natural language understanding and machine learning, we can hope to provide better human-computer collaboration on the broad range of problems suitable for solving with constraint-based reasoning.

Acknowledgments

This publication has emanated from research supported in part by a grant from Science Foundation Ireland under Grant number 12/RC/2289-P2 at Insight the SFI Research Centre for Data Analytics at UCC, which is co-funded under the European Regional Development Fund.

References

Belaid, M.-B.; Belmecheri, N.; Gotlieb, A.; Lazaar, N.; and Spieker, H. 2022. GEQCA: Generic Qualitative Constraint Acquisition. In Proceedings of the AAAI Conference on Artificial Intelligence 36(4): 3690-3697. Palo Alto, CA: AAAI Press. doi.org/10.1609/aaai.v36i4.20282.

Bessiere, C.; Coletta, R.; Daoudi, A.; Lazaar, N.; Mechqrane, Y.; and Bouyakhf, E. 2014. Boosting Constraint Acquisition via Generalization Queries. In Proceedings of the 21st European Conference on Artificial Intelligence: 99 - 104. Amsterdam, the Netherlands: IOS Press. doi.org/10.3233/978-1-61499-419-0-99

Bessiere, C.; Koriche, F.; Lazaar, N.; and O'Sullivan, B. 2017. Constraint Acquisition. *Artificial Intelligence* 244: 315-342. doi.org/10.1016/j.artint.2015.08.001.

Bessiere, C.; Carbonnel, C.; Dries, A.; Hebrard, E.; Katsirelos, G; Narodytska, N; Quimper, Kostas Stergiou, K; Tsouros, D; and Walsh, T. 2023. Learning constraints through partial queries. *Artificial Intelligence* 319. doi.org/10.1016/j.artint.2023.103896.

Bleukx, I.; Devriendt, J.; Gamba, E.; Bogaerts, B.; and Guns, T. 2023. Simplifying Step-Wise Explanation Sequences. In 29th International Conference on Principles and Practice of Constraint Programming: 11:1–11:20. Germany: Dagstuhl Publishing. doi.org/10.4230/LIPIcs.CP.2023.11.

Bogaerts, B.; Gamba, E.; Claes, J.; and Guns, T. 2020. Step-Wise Explanations of Constraint Satisfaction Problems. In Proceedings of the 24th European Conference on Artificial Intelligence: 640-647. Amsterdam, the Netherlands: IOS Press. doi.org/10.3233/FAIA200149.

Ceugniet, X. 2019. A Cognitive Modeling Assistant to Optimize Complex Decisions. Invited talk slides presented at CP 2019 Workshop on Progress Towards the Holy Grail. Stamford, Connecticut, September 30.

Daoudi, A; Lazaar, N.; Mechqrane, Y.; Bessiere, C.; and Bouyakhf, E. 2015. Detecting Types of Variables for Generalization in Constraint Acquisition, In Proceedings of the IEEE 27th International Conference on Tools with Artificial Intelligence: 413-420. New York, IEEE. doi.org/10.1109/ICTAI.2015.69.

Daoudi, A.; Mechqrane, Y.; Bessiere, C.; Lazaar, N.; and Bouyakhf, E. 2016. Constraint Acquisition with Recommendation Queries. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence: 720-726. Palo Alto, CA: AAAI Press / International Joint Conferences on Artificial Intelligence.

Dev Gupta, S.; Genc, B.; and O'Sullivan, B. 2021. Explanation in Constraint Satisfaction: A Survey. In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence: 4400-4407. CA: International Joint Conferences on Artificial Intelligence. doi.org/10.24963/. .2021/601

Dimosthenis C.; Tsouros, D.; Stergiou, K.; and Bessiere, C. Structure-Driven Multiple Constraint Acquisition. 2019. In Proceedings of the 25th International Conference on Principles and Practice of Constraint Programming: 709–725. Switzerland: Springer Cham. doi.org/10.1007/978-3-030-30048-7_41.

Freuder, E., and Wallace, R. 2002. Suggestion Strategies for Constraint-Based Matchmaker Agents. *International Journal on Artificial Intelligence Tools* 11(1): 3-18. doi.org/10.1142/S0218213002000769.

Freuder, E.; Wallace, R.; Nordlander, T.; and Nordlander, T. 2009. Debugging Constraint Models with Metamodels and Metaknowledge. Paper presented at the CP 2009 Workshop on Constraint Modelling and Reformulation. Lisbon, Portugal, doi.ort/10.13140/2.1.2988.6725

Guns, T.; Gamba, E.; Mulamba, M.; Bleukx, I.; Berden, S.; and Pesa, M. 2023. Sudoku Assistant – an AI-Powered App to Help Solve Pen-and-Paper Sudokus. In Proceedings of the AAAI Conference on Artificial Intelligence 37(13): 16440-16442. Washington, DC: AAAI Press. doi.org/10.1609/aaai.v37i13.27072.

Huard, S., and Freuder, E. 1993. A debugging assistant for incompletely specified constraint network knowledge bases. *International Journal of Expert Systems* 6I(4): 419–446.

Kiziltan, Z.; Lippi, M.; and Torroni, P. 2016. Constraint Detection in Natural Language Problem Descriptions. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence: 744–750. Palo Alto, CA: AAAI Press / International Joint Conferences on Artificial Intelligence.

Mackworth, A. On Reading Sketch Maps. 1977. In Proceedings of the Fifth International Joint Conference on Artificial Intelligence: 598-606. AAAI Press / International Joint Conferences on Artificial Intelligence. CA: IJCAI Organization.

Nordlander, T.; Freuder, E.; and Wallace, R. 2007. Maintaining constraint-based applications. In Proceedings of the 4th International Conference on Knowledge Capture: 79-86. New York: ACM.

Prestwich, S. 2020. Robust constraint acquisition by sequential analysis,. In Proceedings of the European Conference on Artificial Intelligence,: 355-362. Amsterdam, the Netherlands: IOS Press. doi.org/10.3233/FAIA200113.

Rossi, F., and Sperduti, A. 2004. Acquiring Both Constraint and Solution Preferences in Interactive Constraint Systems. *Constraints* 9: 311-332. doi.org/10.1023/B:CONS.0000049206.43218.5f.

Schekotihin, K., and Friedrich, G. 2009. Argumentation Based Constraint Acquisition. In Proceedings of the Ninth IEEE International Conference on Data Mining: 476-482. New York: IEEE.

Simonis, H. 2023. Requirements for Practical Constraint Acquisition. Paper presented at AAAI 2023 Bridge on Constraint Programming and Machine Learning. Washington, DC, February 7.

Sleeman, D., and Chalmers, S. 2006. Assisting Domain Experts to Formulate and Solve Constraint Satisfaction Problems. In Proceedings of the International Conference Knowledge Engineering and Knowledge Management: 27-34. Berlin: Springer.

Tsouros, D., and Stergiou, K. 2021. Learning Max-CSPs via Active Constraint Acquisition. In Proceedings of the 27th International Conference on Principles and Practice of Constraint Programming: 54:1–54:18. Germany: Dagstuhl Publishing. doi.org/10.4230/LIPIcs.CP.2021.54

Tsouros, D.; Verhaeghe, H.; Kadiouglu, S.; and Guns, T. 2023. Holy Grail 2.0: From Natural Language to Constraint

Models. Paper presented at the CP 2023 Workshop on Progress Towards the Holy Grail. Toronto, Canada, August 27. doi.org/10.4230/LIPIcs.CP.2021.54.