The Big Deal: Applying Constraint Satisfaction Technologies Where It Makes the Difference

Yehuda Naveh

IBM Research – Haifa, Haifa University Campus, Haifa 31905, Israel
naveh@il.ibm.com

Abstract. In my talk, I will present a few industrial-scale applications of satisfaction technology (constraint programming and satisfiability), all of which are of prime importance to the respective business. The talk will focus on high-end solutions to unique but immense problems. This, as opposed to off-the-shelf solutions which are suitable for more commoditized problems. I argue that the former case is where cutting-edge satisfaction technology can bring the most significant impact. The following is an extended abstract of my talk.

1 Introduction

Constraint satisfaction technologies, including satisfiability (SAT) and its younger sibling constraint programming (CP), have fascinated the computer science community for decades. One of the most intriguing aspects is their declarative nature, bridging the gap between the front-end specification of the problem, and the back-end algorithm which solves it. Thus, the unique position of the discipline at the crossways of artificial intelligence, programming models, logic, algorithms, and theory accounts for much of the charm of this area.

Furthermore, the declarative nature of constraint satisfaction is also the basis of its strong practical importance, and provides the linkage to operations research areas, in particular linear and non-linear optimization. The ability of the user to specify the problem in a language which emerges from the actual application domain may be of critical importance, especially in domains which are complex, dynamic, and require a fast response.

The purpose of my talk is to present a few application domains which exhibit those criteria, and to show how constraint satisfaction is applied in these domains. The common theme of all applications I will describe is their immensity. In fact, all applications have only a small number of instances worldwide (for example, there are only a few high-end truck manufacturers in the world). However, each such instance is of a huge strategic importance to its company. My claim is that it is those cases in which it is beneficial for the company to invest a large effort (or a large amount of resources) in building a high-end constraint-satisfaction-based solution. This, in contrast to the more traditional operations research solutions, which are of a more commoditized nature, and which serve to solve problems more commonly exhibited in many small and medium sized businesses.

O. Strichman and S. Szeider (Eds.): SAT 2010, LNCS 6175, pp. 1–7, 2010.

2 Hardware Verification

Hardware verification is perhaps the prime example of the application of satisfaction technologies at a huge industrial scale. Here, the goal is to ensure that a hardware design works according to its specification while still at the design phase, before cast into silicon. The goal is so important that all large hardware manufacturers, as well as electronic design tool vendors, have for years invested a large amount of resources in the R&D of satisfaction technologies for this domain.

2.1 Model Checking

Model checking [1] for hardware verification is beyond the scope of this talk because the audience is intimately familiar with this topic. It is by far the largest and most important industrial application of SAT technology.

2.2 Stimuli Generation

While model checking and other formal verification techniques have their clear advantages, most notably the ability to formally prove functional correctness of the design, they can hardly cope with modern complex designs at the level of a single unit or larger. To this end, simulation-based verification, in which the design behavior is checked by simulating it over external inputs, accounts for roughly ninety percent of the overall verification efforts and resources.

The major challenge in such methods is in creating inputs, or 'stimuli', which are (1) valid according to the hardware specification and the simulation environment, (2) interesting in the sense that they are likely to excite prone to bugs areas of the design, and (3) diverse [2,3].

Item (1) is dealt with by modeling the entire hardware specification, as well as that of the simulation environment, as a set of mandatory constraints over the simulated variables (memory addresses, data transferred, processor instruction parameters, and so on). Item (2) is dealt with in two ways: first, generic expert knowledge is modeled as a set of soft, non-mandatory, constraints (for example, a soft constraint may require the result of operation $a+b$ to be zero, because this is a known prone-to-bugs area of the floating point processing unit). In addition, the verification engineer, who is directly responsible for creating the stimuli, may add mandatory and non-mandatory constraints to any particular run, directing the stimuli into required scenarios.

Figure 1 illustrates this scheme by considering two variables: the effective address and the real address of a 'load' instruction. These two variables are related by architectural constraints (complex translation scheme), expert knowledge (requirement to reuse cache rows), and specific verification scenarios.

Once modeled, this set of mandatory and non-mandatory constraints can be fed into a constraint solver, which comes up with a solution to the constraint problem in the form of a valid and interesting stimulus. In order to achieve target item (3), the solver typically has a built-in diversification mechanism,

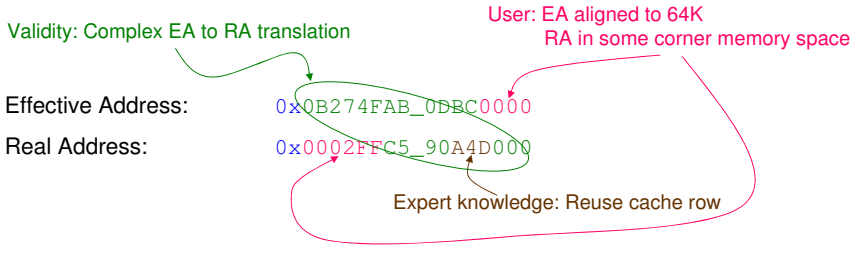

Fig. 1. Stimuli generation constraints on effective and real address values

meaning that each time it is called on the same input (same set of constraints), a sufficiently different output (stimuli) is returned.

3 Workforce Management

Not many companies have a professional services workforce on the scale of tens of thousands of professionals. However, those who are in this business face a critical challenge [4]. How do they identify and assign a team of professionals who best fit a specific customer engagement? Each of the professionals in the team must be skilled — but not over-skilled — to do the job, must be available at the area, or otherwise able to work remotely, must be free of their current engagement and not committed to further projects before the expected end of the work, and must have a personal affinity for the job. The team as a whole must have the correct distribution of skills and of experience levels, must conform to the budget requirements, and needs to be composed of professionals able to work together with each other. In addition, at any given time we need to staff as many projects as possible.

All those 'musts' and 'needs' better be met, or the projects would suffer the consequences of assigning under-qualified or over-qualified teams. These consequences include prolonged project durations, excessive compensation costs, fines and interests, unnecessary commute, and disruption to other projects. In addition to monetary losses, a poorly staffed project may result in an unsatisfied customer and demotivated professionals, leading eventually both to customer churn and employee attrition, which may become a death stroke for the business.

The above problem translates into a constraint problem, where some of the constraints have a clear mathematical foundation, while others are softer in the sense that they describe human attributes and as such are handled in a less strict manner.

The most obvious example of the first kind of constraints is that the same professional cannot be assigned to two different projects if the projects overlap in time. This consideration leads to a new type of global constraint, the `some-different` constraint [5]. The second type of constraints is best dealt

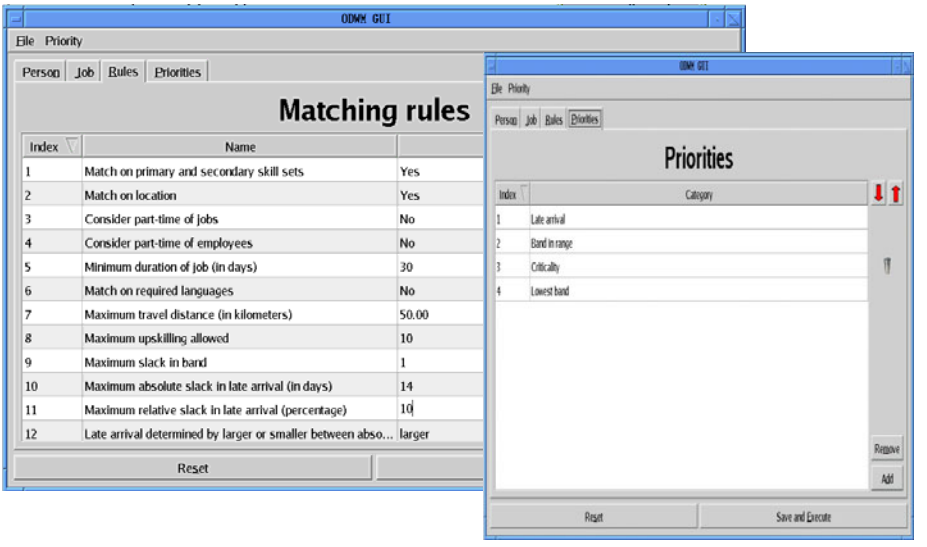

Fig. 2. User interface for specifying hard matching rules and soft priorities for the workforce management problem of service professionals

with as a set of preferences. For example, is it more important for the business to engage professionals with the exact required skill levels, or should geographical proximity of the professionals be the prime concern? similarly, is it more important to have best fits to individual projects, or is it better to maximize the overall number of engagements at the cost of compromising each individual engagement? These sets of rules and preferences are defined by the user in a list as in Fig. 2, and is then translated into a set of hard and soft constraints, respectively, which in turn are solved by a constraint solver.

The ability of constraint programming to account for the rigid mathematical constraints at the same level with the soft human aspects of the problem, is what provides for the unique business advantage of this approach.

4 Truck Configuration

Unlike regular automobiles, for which we enter our local neighborhood dealer and once we decide on a model we get a few options to choose from, large trucks, which cost a few hundred thousand dollars a piece, are highly configurable according to specific business needs of the customers. In fact, unless ordered as a batch from a single customer, there are no two identical large trucks on the road.

The customer, when ordering a truck, has some very specific needs in mind, and will not happily compromise on them. For example, a large dairy company may require a cargo area able to reach a specific temperature while driving in the desert, a drive-train suitable for mountainous terrain, a cabin with space and accessories to meet agreements with the drivers union, and a position for a

Jack is either telescopic 12 T, or regular 25 T
If fuel prefilter is with heated water separator, air-intake cannot be behind cab with round filter
If front axle design is straight, then front axle weight is 7.5 T, and there is no front override guard

Fig. 3. Examples of rules which must be met by any truck configuration

crane of up to half-ton leverage. Those requirements need to be all met, while still conforming to a multitude of engineering, manufacturing, marketing, and legal constraints. An inability to satisfy the customer's needs may result in the customer ordering the truck from a competitor. Conversely, conforming to the specified requirements, may lead to a happy customer and another million-dollar deal.

Given the thousands of configurable variables, and the tens of thousands of constraints on those variables (see some examples in Fig. 3), and given that the configuration problem is NP-complete, it may well be the case that a valid configuration which satisfies the customer's requirements exists, but is not found by the configurator at hand. Therefore, this is a classic example where a stronger technology, incorporating the best algorithms and heuristics available, can truly make the difference.

5 Systems Engineering

Complex systems design (the canonical example is that of an airplane) involves many different disciplines such as requirements engineering, system architecture, mechanical engineering, software engineering, electronic engineering, testing, parametric analysis, and more. In each of these disciplines a model of the product is managed, see Fig. 4. Today all of those models are managed separately. At best, there is an integration of two models, usually done by simple copy or export. This limits the possibilities to maintain traceability between the different teams and project parts, allow synchronization of the data, perform impact analysis when part of the model changes, and achieve optimal design.

One of the major obstacles limiting the ability to combine all sub-models into a single coherent model is the complexity of creating and maintaining a valid and consistent structure. The issue is the various validity rules which the models must conform to (for example, a given methodology may require that each functional feature must be associated with at least one, but not more than three, tests). Attempts to link the various models together may very fast lead to inconsistencies with respect to those rules.

Constraint satisfaction technology can ensure that the links are created in accordance with the design methodology, detect discrepancies between the

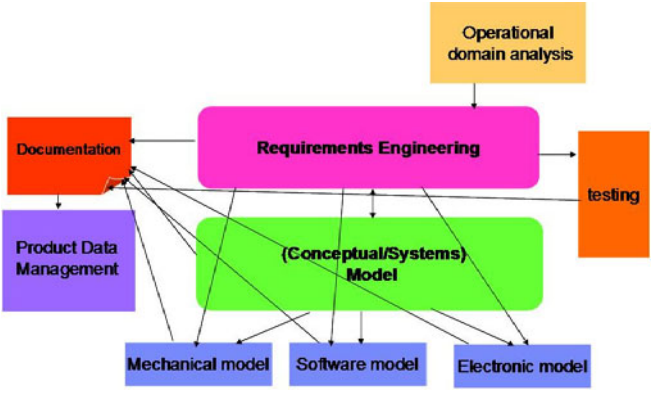

Fig. 4. Inter-relations between models of different disciplines in systems engineering

models, and deduce the existence or absence of links, thus assisting the creation of a unified model.

6 Additional Areas

Above I discussed thriving applications of satisfaction technologies. Space limitation allowed only mere description of the problems, and some flavor of their criticality to the business. It also forbids me from detailing other applications of similar nature, but which are currently only at various levels of prototyping. These include placement of virtual machines on physical hosts at data centers, job-scheduling for massively parallel processors (also known as supercomputers, e.g., IBM's BlueGene), pricing of services engagements, and variability management of product lines.

7 Conclusions

I hope to have delivered the message that in cases where a main part of the business is at stakes, companies would rightfully be willing to invest a large effort in R&D, and specifically in constraint satisfaction technologies. This is the case in hardware verification (shipping functionally correct hardware to the market), services project staffing (ensuring assignments of the correct professional teams), truck configuration (supplying the customer with the truck they need), systems engineering (better management of complex, airplane-size, models), and more. In all those huge-application cases, constraint satisfaction technology can be the means to achieve better, cutting edge, results, and thus provide the competitive advantage at the most critical aspect of the business.

Epilogue and Acknowledgments

The applications described above are all part of IBM's current activities. Obviously, only a strong team of motivated and experienced researchers can reach such achievements. The model at IBM Research is to develop a generic solver as a single technological core, while each application is then developed as an independent module using the same core solver as others. When an application requires an improved algorithm, the application and solver teams together find the right level of generalization of the requirements, and it is then implemented in the solver. This way, all other applications benefit from the original application's needs.

The number of past and present researchers who contributed to this work in the past fifteen years is too large to enumerate here. Many contributors to the stimuli generation activities can be found as authors of the references in [2,3]. As for the other applications discussed, the contributors generally come from the Constraint Satisfaction group at IBM Research – Haifa. The group consists of the solver team: Merav Aharoni, Ari Freund, Wesam Ibraheem, and Nathan Fridlyand; Optimatch (workforce management) team: Sigal Asaf, Michael Veksler, and Haggai Eran; truck configuration: Yael Ben-Haim; and systems engineering: Odellia Boni. In addition, Mage, our independent SAT Solver, is developed by Tanya Veksler and her team at the Formal Verification group. I thank all those contributors who are at the basis of this talk. I am also grateful to Odellia for her writeup of the System Engineering Section of this work.

References

1. Biere, A., Cimatti, A., Clarke, E.M., Fujita, M., Zhu, Y.: Symbolic model checking using sat procedures instead of bdds. In: DAC 1999: Proceedings of the 36th annual ACM/IEEE Design Automation Conference, pp. 317–320. ACM, New York (1999)
2. Naveh, Y., Rimon, M., Jaeger, I., Katz, Y., Vinov, M., Marcus, E., Shurek, G.: Constraint-based random stimuli generation for hardware verification. AI Magazine 28, 13–30 (2007)
3. Adir, A., Naveh, Y.: Stimuli generation for functional hardware verification with constraint programming. In: van Hentenryck, P., Milano, M. (eds.) Hybrid Optimization: the 10 Years of CPAIOR (to appear, 2010)
4. Naveh, Y., Richter, Y., Altshuler, Y., Gresh, D.L., Connors, D.P.: Workforce optimization: Identification and assignment of professional workers using constraint programming. IBM Journal of Research and Development 51, 263–280 (2007)
5. Richter, Y., Freund, A., Naveh, Y.: Generalizing alldifferent: The somedifferent constraint. In: Benhamou, F. (ed.) CP 2006. LNCS, vol. 4204, pp. 468–483. Springer, Heidelberg (2006)