Solving Configuration Problems in Excel

Lucas Bordeaux and Youssef Hamadi
Microsoft Research Ltd.,
7 J J Thomson Avenue Cambridge CB3 0FB,
United Kingdom,
{lucasb, youssefh}@microsoft.com

Abstract

This work shows how Excel can be leveraged to tackle configuration problems. We demonstrate this by presenting the full development cycle of a PC-configurator prototype. We articulate our demonstration by exemplifying several important points. First, Excel can be used to import raw products and services data. Second, these data can be used by a constraint modelling to correctly instantiate a set of configuration rules. Third, any update to the data like addition of new components can be performed without altering the modelling. Fourth, dedicated Excel GUIs can be used for interactive product and service configuration.

Keywords: Excel, Office, Configuration problems, B2B/B2C

Introduction

Product configurators are helping salespersons and partners to successfully match customers' requirements to unique products and service offerings. These tools are critical since they greatly reduce the complexity and therefore improve productivity. We can distinguish between several configuration problem classes : ship-to-order (STO), assemble-to-order (ATO), engineer-to-order (ETO), and Internet Pricing and Configuration (IPC).

STO products have little variability other than a predetermined set of attributes, such as color and size. Some office equipment, household appliances and televisions fall into this classification. ATO products are configurable offerings made up of standard components. They are configured based on customer needs and intercomponent attribute relationships (such as compatibility). Computers and telecommunications equipment fall into this classification. ETO products have the same features as ATO products, but involve some level of engineering analysis to be configured. Finally, non-product based configuration like IPC enables selling channels to deploy customized trade promotions and implement complex pricing and discounting strategies (Desisto 2006).

In order to be successful, product/service configuration must rely on a model that enables users to efficiently enter their needs, integrates with other entreprise systems, and generates outputs such as a line item in a quotation or a bill of material for manufacturing processes. Technically, this model is used by an automated tool like a Constraint Solver[1] whose algorithms are used to perform proactive consistency check of the modelling with respect to the select and deselect actions performed by end-users. Moreover these algorithms can also automatically "finish" an order by completing remaining choice at any time during the interaction. When cost or quality data are available, optimization algorithms can automatically perform the choices which minimize some clearly defined cost function e.g., components with cheapest overall cost in ATO, percentage choice which minimizes duration of financing plans in IPC, etc.

In this paper, we claim that Microsoft Excel has all the required flexibility to implement efficient configurators. To support this claim we are going to demonstrate various points. The most critical being the programmatic integration of a Constraint Solver within Excel. Here the goal is to demonstrate that constraint modelling can effectively be performed without violating Excel's programmatic model. We also need to show that raw business data which characterize products and services can easily be integrated and addressed by Excel's Constraints modelling. Finally, we must show that classical interactive configuration is possible through Excel GUIs.

The paper is articulated as follow. The core is made of an example of configuration problem modelled and solved using a version of Excel which is enhanced with a Constraint Programming (CP) plug-in (Section); we then briefly present some other possibilities of such a plug-in in Section , and give concluding remarks in Section .

Solving Configuration Problems in Excel

We describe here our current prototype which integrates the MSRC Constraints Solver, Disolver in Excel (Hamadi 2003). For the ease of presentation, we consider a classical "PC Configuration" benchmark widely used in the configuration research community (Cli), and which falls under the

[1] Constraint solvers should not be confused with linear programming (LP) solvers such as the one which is readily available in Excel; the tools share similarities but constraint programming deals with more general types of constraints, such as Boolean constraints, which cannot be expressed naturally in LP and are needed to model business rules (see Section).

ATO class. As its name suggests, this problem concerns the configuration of PCs which can be built using several types of processors, mother boards, graphic cards and other components. Some components are incompatible, for instance a mother board of type "Abit BX6 ATX" is compatible with a "Slot1" CPU slot but not with other types of CPUs, *etc*. The goal is to find a configuration which is compatible with the desiderata of the user. The application helps in two ways: (1) when the user imposes a choice for a particular component, or forbids some values for this component, the choices for other components which become incompatible are automatically discarded; this is the *propagation* mode; (2) when the user has specified all her desiderata, she can click a "solve" button and the application will automatically propose her with a fully-specified solution satisfying her requirements (for instance, if the user did not specify any preference regarding the bus component, the application will fill-in the corresponding cell with a value consistent with the other choices); this is the *resolution* mode.

Data Integration

An efficient configurator must provide an easy way to integrate new data such as new product components which should be entered without altering the configuration modelling and the presentation layer parts.

Excel can easily integrate raw data and present them in a column-based form.

Configuration Modelling

In Constraint Programming, modelling involves the definition of constraints restricting the possible combinations of the variables of a problem. Therefore we need to show that we can easily express CP variables in Excel and easily restrict their range through constraints.

Decision variables A discrete decision variable represents a set of alternative choices which will be eventually pruned through constraint propagation and/or end-users decisions. For instance a variable can represent the *motherboard* component and its domain can be made of potential choices, {*AopenAK-72133ATX*, *AsusK7MATX*, *Microstar6167ATX*, *etc.*}.

A new function can be added to Excel to define a decision variable. This function among other things takes as input a range of cell populating its domain.

For example,

```
= CP_Var("MotherboardI",
'PC components'!H:H)
```

defines a variable called "MotherBoard" (this is just an internal name useful for data output) and populates its domain with components data located in $H : H$. Here we can remark that the integration of new mother board components can be made without altering the modelling since the addition of new data in $H : H$ will be seamlessly reported in the related variable domain.

This variable and all the decision variables of the PC configurator benchmark are defined in the *Variables* section of the worksheet presented in figure 1.

Since a variable is represented as a function it returns its value. This value essentially encodes the result assigned to the variable. Depending on which mode is being used (*i.e.* propagation or resolution), variables may be all instantiated or some of them may have a range of possible values. In this case, several choices are obviously possible: we can either display the range of allowed values (*e.g.* {*AopenAK-72133ATX*, *AsusK7MATX*}), or display a single proposal of value (emphasized with a special colour, so that the user sees that other values are possible for this variable), or simply display a special flag indicating that the variable is not instantiated yet.

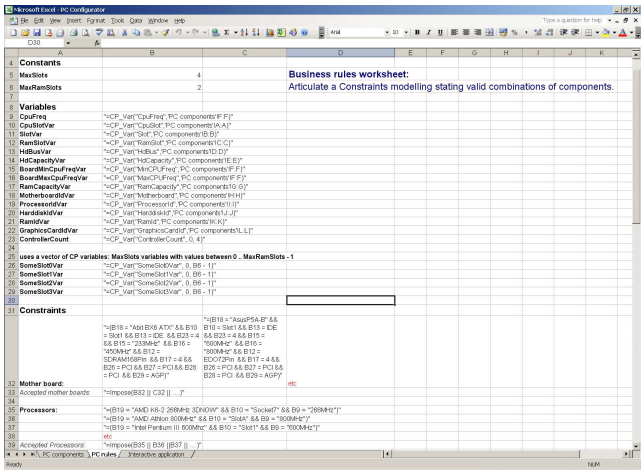

Figure 1: Definition of the configuration modelling through access to the Constraints solver components

Business rules We know how to define the decision variables which will encode valid configurations. We now need to define constraints which will represent the business rules expressing the correct combinations of decision variables.

A business rule is defined as a new Excel function which takes as input a collection of decision variables and maintains a specific invariant on them e.g., all the variables must have different values. As we have presented, CP variables are located in Excel cells and can therefore easily become the input of rules expressed as new Excel functions which will add constraints to the Disolver model. This is presented in the Constraints part of figure 1.

The first rule is related to the mother board components. It defines the list of accepted configurations through a list of conjunctions (presented in line 32). Each conjunction accesses decision variables and specifies a correct mother board realisation. For instance, we find in cell B32,

```
= (B18 = "Abit BX6 ATX" && B10 = Slot1
&& B13 = IDE && B23 = 4 && B15 = "233MHz"
&& B16 = "450MHz" && B12 = SDRAM168Pin
&& B17 = 4 && B26 = PCI && B27 = PCI
&& B28 = PCI && B29 = AGP)
```

This rule validates a configuration using an "Abit BX6 ATX" mother board combined to a "Slot1" CPU slot, etc.

The 13 possible mother board configurations of the PC benchmark can be defined with as many conjunction constraints ultimately combined in a disjunctive formula. This is presented in B33,

```
= Impose(B32 || C32 || ...)
```

This constraint imposes a disjunction between rules B32, C32, etc[2]. It takes as input the previous set of rules. Remark that since a rule is represented as a function it returns its truth value[3] which can be reused by some other rule, or imposed in a constraint. Valid Processor components are similarly defined in the bottom of figure 1 through a simple disjunction of conjunctions.

Remark that we do not give all the rules of this model which also contains conjunction of disjunction which specify correct combinations of all the remaining parts of a proper PC.

Interactive application Excel GUIs can easily use our configuration modelling to provide an interactive front-end to end-users. A typical GUI is made of several combo-boxes which report the domains of the decision variables. Each select or deselect action is reported to the modelling where business rules are enforced through constraints and eventually lead to value suppressions in other decision variables.

Combo-boxes can be addressed in any order with the insurance of moving in a consistent space of configurations.

At any time, a *Solve* button can be pressed to finish a partial configuration. In optimization settings we can imagine several Solve button related to different optimization criterion, e.g., Solve/cheapest, etc.

Additionally, an interaction can be saved for later access through appropriate buttons.

Related and Future Work

Incorporating a constraint solver in a spreadsheet is not new[4]. Like (Felfernig *et al.* 2003), we think that these additions are not useful in general but can only benefit to some problem classes like simple configuration problems.

Configuration problems are important and as it was demonstrated, they can be solved in Excel through the integration of a Constraints Solver. But, as we are going to see, integrating a Constraints Solver in Excel opens new opportunities.

Dependencies between data in spreadsheets are currently functional dependencies. For instance we can specify that cell *Benefits!C5* is equal to the sum of all numbers of column F in spreadsheet *February04*. Modifying the value of one cell in column F will propagate to cell $C5$.

On the other hand, there is no way to propagate the dependencies backward and to automatically deduce, for instance, values or intervals of values for the cells of column F which are not filled yet, given a prospective value for $C5$. This functionality would be of considerable interest since it would allow the user to interactively simulate how modifications in some of her data impact other data, to see whether some combinations are possible and to determine which optimal values are possible for some cells. This appears to match the way many customers currently use Excel - but so far all they can rely on is a unidirectional form of propagation.

Therefore, an interesting new possibility would be to add to Excel ways to express richer data dependencies than the current functional ones: relational dependencies.

Conclusion

In this paper, we showed how Excel can be leveraged for configuration problems. Beside the simple addition of a new capability to a flagship product, we think that the proposed extension would be a nimble way to address B2B and B2C markets. Indeed, Office is pervasive and any Office-based Business Solution would become widely adopted by a large basis of Office customers. Because Excel is so popular, we believe that our proposal could indeed contribute to a wider adoption of configuration technologies, since it is conceivable that more users would get familiar with it once it is available in the Office package, and would then get interested in more specialised offers as their use of configuration tools increases.

References

Clib:configuration-benchmarks-library. http://www.itu.dk/research/cla/externals/clib/.

Desisto, R. 2006. Marketscope for sales configuration, 2Q06. Technical Report n/a, Gartner.

Felfernig, A.; Friedrich, G.; Jannach, D.; Russ, C.; and Zanker, M. 2003. Developing constraint-based applications with spreadsheets. In *IEA/AIE*, 197–207.

Hamadi, Y. 2003. Disolver : A Distributed Constraint Solver. Technical Report MSR-TR-2003-91, Microsoft Research.

[2]One way to see that is that we encode each table of allowed combinations between components using a disjunction of tuples, each of which is represented as a conjunction of basic equalities. Alternatively, it would be possible to use a different CP mapping and to directly use Excel tables, mapped into Table constraints.

[3]Note also that this value is not necessarily fully specified if we are in propagation mode. In this case, one option is to return the set $\{0, 1\}$, reflecting the fact that both values 0 (false) and 1 (true) are possible for the rule.

[4]See for instance, http://bach.istc.kobe-u.ac.jp/cream/calc.html.