

From Business Rules to Constraint Programs in Warehouse Management Systems

Student: Julien Martin
Supervisor: François Fages

Projet Contraintes, INRIA Rocquencourt, France

Abstract. This thesis aims to provide a knowledge representation language for Packing problems allowing to state higher-dimensional placement constraints with business rules, and to translate them into constraint programs. This paper presents a three-level hierarchy of knowledge representation languages for Packing problems: geometrical constraints, placement constraints and business rules. The challenge is to translate business rules into efficient constraint programs in this domain. We describe our first implementation in SICStus prolog and Constraint Handling Rules.

1 Introduction

The purpose of a Warehouse Management Systems (WMS) is to control movement and storage of materials within a warehouse. Generally, WMS aim to provide optimization functionalities like scheduling, advanced packing tools, optimal filling of containers subject to delivery constraints, *etc.* The increasing demand on more numerical handling necessitates to master the complexity of several NP-hard problems such as: how to pack items in a container, how many cartons are needed to pack customer items, how to schedule the manpower so as to finish the preparation in time, in which order to pack items in a pallet, or how to place pallets in a truck according to the customers to visit.

Rules for knowledge representation are considered in many industry fields because they allow to express clearly, flexibly and concisely the expertise related to the processes of an activity, of a business. It should be noted that such rules express two kinds of knowledge: a declarative knowledge that gives information about what a solution must satisfy, and a procedural knowledge that provides information about how to build a solution.

The main objective of this thesis is to come up with business rules that generate a constraint program in order to find solutions that respect the declarative knowledge of a business activity. To date, there is no general scheme for translating business rules into constraint programs, even in limited domains where the state-of-the-art is rather in the filling of predefined problem patterns. In the framework of the Net-WMS European Union STReP project¹, the focus is more particularly on the Container Loading (or Packing) problem: to pack a set

¹ <http://net-wms.ercim.org/>

of parallelepipeds of various sizes and weights into one container such that the wasted volume of the container is minimised, with respect to a set of constraints.

The structure of this paper is based on a hierarchical arrangement of knowledge representation languages that we propose. Section 2 describes a minimal constraint language permitting to express any constraint between simple geometrical objects. Section 3 presents a higher-level constraint language made up of so-called placement constraints. An example of translation of these constraints into geometrical ones is given. Section 4 introduces the highest level of knowledge representation that consists of business rules. According to the same principle, the translation of simple Packing business rules into placement constraints is presented. A prototype implementation in Constraint Handling Rules [7] and SICStus Prolog is then briefly described, before we finally conclude on our current directions of research for improving the performance of this scheme.

2 Geometrical Constraints

The most immediate way to state topological constraints between three-dimensional objects is to assert inequalities between their different one-dimensional projections, *i.e.* their co-ordinates upon a particular dimension. The objects are assumed to be parallelepipeds and the space to be discrete. The geometrical constraints constitute the constraint program knowledge representation level and the vocabulary is the following:

$$\begin{aligned} \text{Functions: } \Sigma_F &= \mathbb{N} \cup \{+, -, \min, \max\} \\ \text{Constraints: } \Sigma_C &= \{<, \leq, =, \geq, >, \neq\} \end{aligned}$$

Furthermore, in addition to conjunction (\wedge), we allow the classical logic connectives ($\vee, \Rightarrow, \Leftrightarrow$) between constraints. In particular, constraints of the form $c \Rightarrow d$, *i.e.* conditional constraints, are of interest because of the structure of business rules. These combinations of constraints can be implemented by reification.

From a novice point of view however, such a low level encoding of Packing problems into geometrical constraints is difficult, especially if complex constraints are needed. Hence the need for higher-level languages to bridge the gap between constraint programming and business experts.

3 Placement Constraints

The placement constraints represent the vocabulary of basic business knowledge of Packing: they form a specialized vocabulary that intends to state any constraint related to the Packing problem paradigm.

A set of binary constraints applying on parallelepipeds in a discrete space was introduced in [1]. This set of constraints constitutes the basis of a prototype language for Packing problems covering the elementary concepts.

Constraints: $\Sigma_C = \{overlap, non_overlap, includes, distance, equal, before, on_top, above, touch, oversize, ori_siz_end, heavier_than\}$

Let \mathbb{P} be the set of parallelepipeds, where a parallelepiped is a 10-tuples of integers representing the origin, the size and the end of its segments in the three dimensions, plus the weight. Let $O_{i,d}$, $E_{i,d}$, be finite domain variables, where $1 \leq i \leq card(\mathbb{P})$ and let $d \in \{1, 2, 3\}$, denoting respectively the origins and the ends of parallelepipeds P_i in the three dimensions $\{x, y, z\}$. The weight and the sizes of parallelepiped P_i are denoted by w_i , $s_{i,d} \in \mathbb{N}$. As in the geometrical language, non-atomic placement constraints are formed with the logical connectives ($\wedge, \vee, \Rightarrow, \Leftrightarrow$).

Most of the placement constraints are fall under two versions:

- the one-dimensional version like $includes(P_1, P_2, D)$ or $overlap(P_1, P_2, D)$, where $D \in \{1, 2, 3\}$ stands for the dimension, which means that the projection of P_1 upon dimension D includes (resp. overlaps with) the projection of P_2 upon the same dimension D .
- the three-dimensional version like for instance $includes(P_1, P_2)$ or $overlap(P_1, P_2)$ which has a meaning in a three-dimensional space.

Placement constraints can be translated into geometrical constraints as follows:

- **before**(P_1, P_2, D): $E_{1,D} \leq O_{2,D}$
- **overlap**(P_1, P_2, D): $(\min(E_{1,D}, E_{2,D}) - \max(O_{1,D}, O_{2,D})) > 0$
- **distance**(P_1, P_2, D, L): $(\max(O_{1,D}, O_{2,D}) - \min(E_{1,D}, E_{2,D})) = L$
- **heavier_than**(P_1, P_2): $w_1 \geq w_2$

It is worth noticing that *on_top* is defined by the conjunction of the two placement constraints *touch* and *before*. Once again, *touch* is defined with the help of the placement constraints *distance* and *overlap* :

- **on_top**(P_1, P_2): $touch(P_1, P_2, 3) \wedge before(P_2, P_1, 3)$
- **touch**(P_1, P_2, D): $\bigwedge_{d \in \{1,2,3\} \setminus \{D\}} overlap(P_1, P_2, d) \wedge distance(P_1, P_2, D) = 0$

The complete translation $\theta(on_top(P_1, P_2))$ into geometrical constraints is:

$$on_top(P_1, P_2) \begin{cases} touch(P_1, P_2) \\ before(P_1, P_2, 3) \end{cases} \begin{cases} \min(E_{1,1}, E_{2,1}) - \max(O_{1,1}, O_{2,1}) > 0 \\ \wedge \min(E_{1,2}, E_{2,2}) - \max(O_{1,2}, O_{2,2}) > 0 \\ \wedge \max(O_{1,3}, O_{2,3}) - \min(E_{1,3}, E_{2,3}) = 0 \\ \wedge E_{2,3} \leq O_{1,3} \end{cases}$$

That translation has a linear complexity in the number of placement constraints.

4 Business Rules of Packing

Although business rules do not have a widely accepted formal definition [4], they are a popular means of expressing knowledge in industry: they are written in natural language, and they can be validated one by one by business people [3]. A business rule can be formalized as an expression of the form: < **Declaration** declaration: **If** condition **Then** action >. The declaration states the objects that are handled within the condition and action parts. A Packing rule is a business rule where facts and terms of its condition and action expressions are restricted to the basic business vocabulary of Packing.

Example 1 (The Basic Stacking Rule).

- "All boxes must be stacked in decreasing order of weight"
- **Declaration** For all boxes P_1, P_2 :
If P_1 is on top of P_2
Then P_2 is heavier than P_1

This rule translates into the placement constraints

- For all boxes P_1, P_2 , $on_top(P_1, P_2) \Rightarrow heavier_than(P_2, P_1)$

Recall that $\theta(on_top(P_1, P_2))$ is the complete translation of the constraint $on_top(P_1, P_2)$ into geometrical constraints, as given in section 3. Thus, the complete translation of the Basic Stacking Rule into conditional geometrical constraints is: $\theta(on_top(P_1, P_2)) \Rightarrow w_2 \geq w_1$. The translation from the Basic Stacking Rule into placement constraints has thus a quadratic complexity in the number of boxes.

There exists another family of rules that involve constraints between sets, like the following one:

Example 2 (The Weight Distribution Rule).

- "Over the container length, the weight supported by one half must not exceed 10% more than the weight supported by the other half"
- **Declaration** For the sets of boxes S_1, S_2 that belong to the set of all subsets of parallelepipeds \mathbb{P} , and for the container C .
If the set of boxes S_1 is included in the left half-length of C ,
and the set of boxes S_2 is included in the right half-length of C
such that the union of S_1 and S_2 equals the set \mathbb{P} ,
Then the maximum weight between S_1 and S_2 divided by
the minimum weight between S_1 and S_2 is less or equal to 1.1

This motivates the extension of the placement constraints language with set constraints [8]. However, in absence of set constraints, such a rule can be handled with reification by the constraint $max(l,r) \leq 1.1 \times min(l,r)$ where $x = \sum_{i=1}^{card(\mathbb{P})} W_i L_i$, $y = \sum_{i=1}^{card(\mathbb{P})} W_i R_i$. and L_i and R_i are boolean variables representing the truth value of the constraint " P_i is in the left (right) half-length of the container".

5 Implementation

In our current implementation, the translation of constraints from one level of knowledge representation to the lower one is done by Constraint Handling Rules (CHR), while the consistency check of the resulting geometrical constraints is ensured by the SICStus prolog $CLP(\mathcal{FD})$ solver [5].

Conditional constraints are handled by constraint reification, like for instance: $B \#<=> (E1 \#=< O2)$, which means that the boolean variable B reflects the truth value of the less or equal constraint between $E1$ and $O2$, *i.e.* its entailment. Then, the variable B can appear in the left side of a conditional constraint. The disadvantage of reification is weak constraint propagation.

Example 3 (CHR mapping of Placement constraints into Geometrical constraints).

<pre> before(P1, P2, D, B) <=> end(P1, D, E1), origin(P2, D, O2), B #<=> (E1 #=< O2). overlap(P1, P2, D, B) <=> origin(P1, D, O1), end(P1, D, E1), origin(P2, D, O2), end(P2, D, E2), B #<=> ((min(E1,E2) - max(O1,O2)) > 0). </pre>	<pre> on_top(P1, P2, B) <=> touch(P1,P2, 3,P1P2tchZ), before(P2,P1, 3,P2bfrP1Z), B #<=> (P2bfrP1Z #/\ P1P2tchZ). </pre>
---	---

Example 4 (CHR mapping of Basic Stacking Rule into Placement constraints).

```

box(P1), box(P2) ==>
  on_top(P1, P2, P1onTopOfP2),
  heavier_than(P2,P1, P2heavierThanP1),
  P1onTopOfP2 #=> P2heavierThanP1.

```

Figure 1 shows a screenshot of our prototype interfaced through CLPGUI [6] with a 3D viewer of Packing solutions and dedicated panels permitting to interact with the solver. Beside interaction capabilities for selecting rules (and

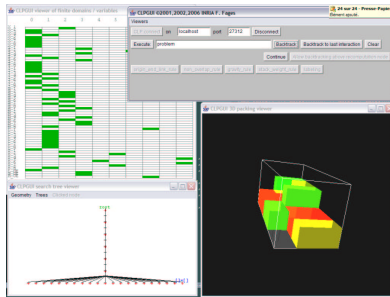


Fig. 1. Visualisation and interaction with the packing solver in CLPGUI.

thus constraints) that must hold or for monitoring the search tree and domains of variables, strategies for modifying a solution and re-optimizing the configuration are under study.

6 Conclusion

A three-level hierarchy of knowledge representation languages for Packing has been presented. The hierarchical structure is accompanied by mappings that allow us to translate, through an intermediate layer of placement constraints, some simple business rules to geometrical constraints, *i.e.* into a constraint program where variables represent co-ordinates taking their value from intervals of \mathbb{N} .

As business rules represent conditional constraints, our first implementation based upon the SICstus Prolog *CLP(FD)* solver makes an intensive use of reification. As a consequence, constraint propagation is quite weak. Moreover, some business rules express set constraints which are not efficiently handled with reified constraints. Future work will thus consider three issues: first, according to the objectives of the Net-WMS project, we will use a new generic global constraint for handling complex objects made of assemblies of parallelepipeds [2], instead of basic geometrical constraints, in order to improve the genericity and the efficiency of our scheme [2]. Second, we will investigate the checking of constraint entailment by a dedicated CHR solver instead of reification, using bound consistency reasoning. Expected result is an earlier pruning of domains and thus a more efficient handling of set constraints. Last, the analysis of business rules for enumeration heuristics will be investigated in connection to the confluence analysis of rule sets [9].

References

1. Beldiceanu N.: Global Constraints as Graph Properties on a Structured Network of Elementary Constraints of the Same Type. In *Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming*, p. 52–66, (2000).
2. Beldiceanu N., Carlsson M.: A Generic Geometrical Constraint Kernel in Space and Time for Handling Polymorphic k-Dimensional Objects. In *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming*, (2007).
3. Business Rules Group: Business Rules Manifesto, (2003)
<http://www.businessrulesgroup.org/brmanifesto/BRManifesto.pdf>.
4. Business Rules Group: Defining Business Rules - What Are They Really? (2000)
http://www.businessrulesgroup.org/first_paper/BRG-what-is-BR_3ed.pdf.
5. Carlsson M., Ottosson G., Carlson B.: An Open-Ended Finite Domain Constraint Solver. In *Proceedings of the 9th International Symposium on Programming Languages: Implementations, Logics, and Programs*, London, p. 191–206, (1997).
6. Fages F., Soliman S., Coolen R.: CLPGUI: A Generic Graphical User Interface for Constraint Logic Programming. *Journal of Constraints, Special Issue on User-Interaction in Constraint Satisfaction*, 9(4):241-262, (2004).
7. Frühwirth T.: Theory and Practice of Constraint Handling Rules. *Journal of Logic Programming*, 37(1-3):95-138, (1998).
8. Gervet C.: Conjunto: Constraint Logic Programming with Finite Set Domains. In *Proceedings of the 1994 International Symposium on Logic programming*, Melbourne, pages 339–358, (1994).
9. Haemmerle R., Fages F.: Abstract Critical Pairs and Confluence of Arbitrary Binary Relations. In *Proceedings of the 18th International Conference on Rewriting Techniques and Applications (RTA'07)*, Paris, (2007).