Abstract

This is the abstract

Contents

1 Acknowledgements

2 Introduction

2.1 Constraint Programming

Constraint programming (CP) is a method to solve problems by first modeling the problem with different types of variables and then adding constraints that the variables must fulfill. The variables have an initial domain of possible values. The CP solver then uses the constraints to remove impossible values. A very typical problem suitable for constraint programming is sudoku. The most intuitive model is a 9x9 matrix of integer variables. The general constraints for all standard sudokus is that every cell has a domain of 1..9, and that all cells in a row, column and box must be different. For a specific sudoku constraints are added to require specific cells to have specific values.

2.2 Gecode

Gecode is an open source CP solver blablabla.

3 Background

3.1 The need for tuple variables

When using a CP solver, one wants to avoid the brute force searching. This is done by intelligently choosing model and constraints. By doing this the variable domains can be pruned to smaller sizes. For example, if two variables x and y have domains $x = \{1, 2, 3\}$ $y = \{2, 3, 4\}$ and the constraint $y < x$ is added we can prune the domains to $x = \{3\}$ and $y = \{2\}$. If we instead have the same variables but add the constraint $x \neq y$ nothing would be pruned, since for all values in both domains there exists solutions. In this example the problem was modeled with two integer variables, but if the problem instead would be modeled with one tuple variable pruning would be possible. Then we would have a tuple variable t with domain $t = \{< 1, 2 >, < 1, 3 >, < 1, 4 >, < 2, 2 >, < 2, 3 >, < 2, 4 >, < 3, 2 >, < 3, 3 >, < 3, 4 >\}$ and pruning with the distinct condition would yield $t = \{< 1, 2 >, < 1, 3 >, < 1, 4 >, < 2, 3 >, < 2, 4 >, < 3, 2 >, < 3, 4 >\}$. This shrinks the size of the search tree from 9 to 7.

If we instead look at the case where $x, y = \{1..1000\}$ and the constraint $x = y$ the benefits is more obvious. Using integer variables pruning can not remove anything. If tuple variables were used, the size of the domain would shrink from 10^6 to 10^3. This indicates that tuple variables can be quite useful.

4 Body

4.1 Implementation

4.2 Performance test

5 Related work

5.1 Implementation of bit-vector variables in a CP solver

5.2 A propagator design framework for constraints over sequences

6 Conclusions and future work

6.1 Write the implementation in such a way that it may be accepted in a new Gecode release

6.2 Test different implementations with different domain representations

6.3 Extend the tuples to arbitrary dimensions and not just pairs

6.4 Reach better performance for a specific problem with an array of pair variables than with two arrays of integer variables

7 Git repository

`http://github.com/patwotrik`

8 Legal

References

[1] Calculus, Adams

[2] Operating systems and its concepts, Galvin