Constraint Relaxation Techniques to Aid the Reuse of Knowledge Bases and Problem Solvers

Tomas Nordlander[1], Ken Brown[2], and Derek Sleeman[1]
[1]Department of Computing Science, University of Aberdeen, Scotland, UK
{tnordlan, sleeman}@csd.abdn.ac.uk
[2]Cork Constraint Computation Centre, Department of Computer Science, University College Cork, Ireland
k.brown@cs.ucc.ie

Abstract: Effective re-use of knowledge bases requires the identification of plausible combinations of both problem solvers and knowledge bases, which can be an expensive task. Can we identify impossible combinations quickly? The capabilities of combinations can be represented using constraints, and we propose using constraint relaxation to help eliminate impossible combinations. If a relaxed constraint representation of a combination is inconsistent then we know that the original combination is inconsistent as well. We examine different relaxation strategies based on constraint graph properties, and we show that removing constraints of low tightness is an efficient strategy which is also simple to implement.

1. Introduction

The MUSKRAT (Multistrategy Knowledge Refinement and Acquisition Toolbox) framework aims to unify problem solving, knowledge acquisition and knowledge-base refinement in a single computational framework [1]. Given a set of Knowledge Bases (KBs) and Problem Solvers (PSs), the MUSKRAT-Advisor [2] investigates whether the available KBs will fulfil the requirements of the selected PS for a given problem. The Advisor informs the user if the available KBs are sufficient. Our research addresses the problem of checking whether combinations of existing KBs could be reused with the selected PS. We propose to represent the KBs and PSs as Constraint Satisfaction Problems (CSPs), which can be combined to produce composite CSPs. If a combined CSP is solvable, then the original combination of KBs with the selected PS could be used to solve the given problem; if the resultant CSP is inconsistent, then the combination cannot be used. Identifying a suitable combination thus requires examining a series of CSPs, and rejecting insolvable ones until we find one with a solution. Proving CSPs insolvable can be a lengthy process; we would like to find a way to do this quickly. The method we propose here is to relax a CSP, and if we can prove that the relaxed version is insolvable then we know that the original CSP does not have a solution either. However, if the relaxed CSP has a solution, then the original CSP represents a *plausible* combination. To test this proposal, we investigate different relaxation strategies for binary CSPs and test them on randomly generated problems. We suggest that removing constraints with low tightness is an effective method for identifying insolvable combinations. Thus this paper reports a contribution to the challenging problem of Knowledge Reuse as it presents an aid based on Constraint Programming to enable a quick identification of inconsistent combinations.

2. Background

This work is supported by the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration, which focuses on six challenges to ease substantial bottlenecks in the engineering and management of knowledge; reuse of knowledge is one of those challenges [3]. Current work in reuse has resulted in systems where a number of components have been reused, including ontologies, problem-solving methods (PSMs), and knowledge bases (KBs) [3]. The use of cases in Case Based Reasoning is a related activity [4].

2.1 Reusing Knowledge Based Systems

One of the main goals of the Knowledge Based System (KBS) community is to build new systems out of existing Problem Solvers (say Problem Solving Methods) and existing Knowledge Bases. At an early stage the Knowledge Engineering sub-area identified a range of Problem Solving Methods, which they argued covered the whole range of problem solving and included methods for Classification and Diagnosis through to Planning (so-called synthesis tasks) [5]. An early but powerful example of reuse of a PSM was the use of the EMYCIN shell with a variety of domain-specific knowledge bases [6]. More recently, systems like PROTÉGÉ have provided an option to write KBs in a standardised format like OKBC [7]. This then takes the goal of building new Knowledge-Based Systems (KBSs) one step further. A possible approach is to take the required KB and PSM and to produce manually, initially, a series of mappings, which will make the 2 components "comparable" [7], and thus to develop a new KBS from pre-existing components.

We have chosen to work with the domain of scheduling, mainly because the constraint community has been successful in addressing real world problems in this area. Also, we argue that the nature of scheduling problems are very close to CSPs, hence it would be relatively easy to transform PSs and KBs in this domain. We will now consider an example of a mobile phone manufacturer to understand our notion of KB and PS reuse. The manufacturer has two factories, three suppliers and two delivery companies to transport phones to wholesalers around the world (Figure 1).

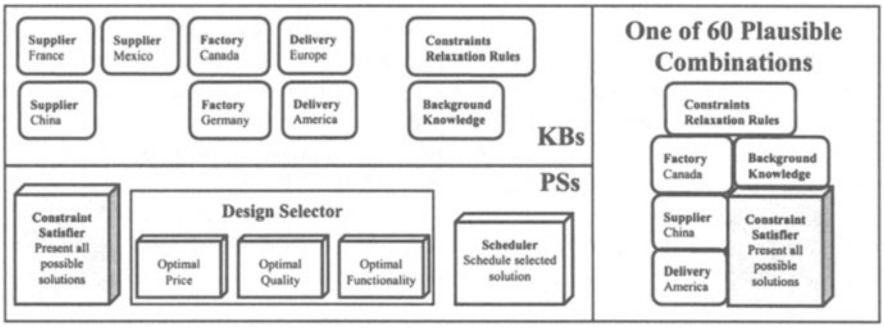

Figure 1. Combining KBs with selected PS

Along with the domain-specific KBs, the system also consists of background knowledge (e.g., ISO 9001, Safety Standards), and constraint relaxation rules to