Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science 185 (2007) 47–62
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

# Incremental Parametric Development of Greedy Algorithms

Dominique Cansell[1],[2]

*LORIA*
*Université de Metz*
*Ile du Saulcy*
*57045 Metz, France*

Dominique Méry[1],[3]

*LORIA*
*Université Henri Poincaré Nancy 1*
*BP 239*
*54506 Vandoeuvre-lès-Nancy, France*

**Abstract**

The event B method provides a general framework for modelling both data structures and algorithms. B models are validated by discharging proof obligations ensuring safety properties. We address the problem of development of greedy algorithms using the seminal work of S. Curtis; she has formalised greedy algorithms in a relational calculus and has provided a list of results ensuring optimality results. Our first contribution is a re-modelling of Curtis's results in the event B framework and a mechanical checking of theorems on greedy algorithms The second contribution is the reuse of the mathematical framework for developing greedy algorithms from event B models; since the resulting event B models are generic, we show how to instantiate generic event B models to derive specific greedy algorithms; generic event B developments help in managing proofs complexity. Consequently, we contribute to the design of a library of proof-based developed algorithms.

*Keywords:* Formal method, B event-based method, refinement, safety, greedy algorithms.

## 1 Introduction

Algorithms provide a class of systems on which one can apply proof-based development techniques like the event B method, especially the refinement. The main

[1] We thank anonymous referees for their comments; we thank Emilie Balland who has developed preliminary models of the study, when she was in the MOSEL team for a training period under the supervision of Dominique Méry. J.-R. Abrial supports our works by friendly messages and comments. Sharon Curtis provides us comments on early developed models.

[2] Email: cansell@loria.fr

[3] Email: mery@loria.fr

doi:10.1016/j.entcs.2007.05.028

advantage is the fact that we teach data structures and algorithms to students, who should have simple explanations of why a given algorithm is effectively working or why some assertion is an invariant for the algorithm under consideration . . . Hence, we have a good knowledge of algorithmic problems and it is simpler for us to apply proof-based development techniques on the algorithmic problems. Greedy algorithms constitute a well defined class of algorithms (applications and properties) and we aim to provide proof-based patterns for facilitating the proof-based development (in B) of greedy algorithms.

In a previous work [4], we have developed Prim's algorithm and we have proved properties over trees: the inductive definition of trees helps in deriving intermediate lemmas asserting that the growing tree converges to the minimal spanning tree, according to the greedy strategy. The resulting algorithm was completely proved using the proof assistant [7] and we can partially reuse current developed models to obtain Dijkstra's algorithm or Kruskal's algorithm. The greedy strategy is not always optimal and the optimality of the resulting algorithm is proved by the theorem 24.1 of Cormen's book [8] in the case of the minimal spanning tree problem. The gain is clear, since we had a mechanised and verified proof of Prim's algorithm. The formalisation of greedy-oriented algorithmic structures was not so complicated but we were assuming that a general theory on greedy structures could help in designing our greedy algorithms using the event B method. Fortunately, S. Curtis [9] brings the theoretical material that was missing in our project; she has formalised in a relational framework properties required for leading to the optimality of solutions, when applying a greedy technique. However, we have not explained why we are choosing the greedy method and what for? Our quest is to propose general proof-based developments (or patterns) for a given problem or for a given paradigm. We think that the refinement provides a way to introduce generic elements in developed models. A second objective is to illustrate the adequacy of the B prover [7], when checking results over set-theoretical structures; in a sense, our work may seem to be a plagiarism of Curtis's paper, but the tool scans each detail to check and it validates each user hint, and, generally, there is no assisted significant proof without human hint (proof step or tricky lemma). Hence, our paper is an exercise in checking properties over greedy structures and in proposing generic development of greedy algorithms; we do not know any other mechanized complete proof-based developments of greedy algorithms.

## *1.1 Greedy algorithms*

Greedy algorithms are used to solve optimization problems like the shortest path problem or the best order to execute a set of jobs. A greedy algorithm works in a local step to satisfy a global constraint. A greedy algorithm can be summarized by the general algorithm 1, where C is the set of candidates and S is the set containing the solution or possibly no solution. The goal is to optimize a set of candidates which is a solution to the problem; the optimization maximizes or minimizes the value of an objective function. The optimization state is checked by the Boolean function called *goodchoice*. Lectures notes of Charlier [6] provide a very complete