Splitting an Operator

An Algebraic Modularity Result and Its Application to Logic Programming

Joost Vennekens, David Gilis, and Marc Denecker

Department of Computer Science, K.U. Leuven
Celestijnenlaan 200A
B-3001 Leuven, Belgium

Abstract. It is well known that, under certain conditions, it is possible to *split* logic programs under stable model semantics, i.e. to divide such a program into a number of different "levels", such that the models of the entire program can be constructed by incrementally constructing models for each level. Similar results exist for other non-monotonic formalisms, such as auto-epistemic logic and default logic. In this work, we present a general, algebraic splitting theory for programs/theories under a fixpoint semantics. Together with the framework of *approximation theory*, a general fixpoint theory for arbitrary operators, this gives us a uniform and powerful way of deriving splitting results for each logic with a fixpoint semantics. We demonstrate the usefulness of these results, by generalizing Lifschitz and Turner's splitting theorem to other semantics for (non-disjunctive) logic programs.

1 Introduction

An important aspect of human reasoning is that it is often incremental in nature. When dealing with a complex domain, we tend to initially restrict ourselves to a small subset of all relevant concepts. Once these "basic" concepts have been figured out, we then build another, more "advanced", layer of concepts on this knowledge. A quite illustrative example of this can be found in most textbooks on computer networking. These typically present a seven-layered model of the way in which computers communicate. First, in the so-called physical layer, there is only talk of hardware and concepts such as wires, cables and electronic pulses. Once these low-level issues have been dealt with, the resulting knowledge becomes a *fixed* base, upon which a new layer, the data-link layer, is built. This no longer considers wires and cables and so on, but rather talks about packages of information travelling from one computer to another. Once again, after the workings of this layer have been figured out, this information is taken "for granted" and becomes part of the foundation upon which a new layer is built. This process continues all the way up to a seventh layer, the application layer, and together all of these layers describe the operation of the entire system.

In this paper, we investigate a formal equivalent of this method. More specifically, we address the question of whether a formal theory in some non-monotonic language can be *split* into a number of different levels or *strata*, such that the formal semantics of the entire theory can be constructed by succesively constructing the semantics of the various strata. (We use the terms "stratification" and "splitting" interchangeably to denote a division into a number of different levels. This is a more general use of both these terms, than in literature such as [Gel87].) Such stratifications are interesting from both a practical and a more theoretical, knowledge representational point of view. For instance, computing models of a stratified version of a theory is often significantly faster than computing models of the original theory. Furthermore, in order to be able to build and maintain large knowledge bases, it is crucial to know which parts of a theory can be analysed or constructed independently and, conversely, whether combining several correct theories will have any unexpected side-effects.

It is therefore not surprising that this issue has already been intensively studied. Indeed, splitting results have been proven for auto-epistemic logic under the semantics of expansions [GP92,NR94] default logic under the semantics of extensions [Tur96] and various kinds of logic programs under the stable model semantics [LT94,EL04]. In all of these works, stratification is seen as a syntactical property of a theory in a certain language under a certain formal semantics.

In this work, we take a different approach to studying this topic. The semantics of several (non-monotonic) logics can be expressed through fixpoint characterizations in some lattice of semantic structures. In such a semantics, the meaning of a theory is described by an operator, which revises proposed "states of affairs". The models of a theory are those states which no longer have to be revised. Knowing such a revision operator for a theory, should suffice to know whether it is stratifiable: this will be the case if no higher levels are ever used to revise the state of affairs for lower-level concepts. This motivates us to study the stratification of these revision operators themselves. As such, we are able to develop a general theory of stratification at an abstract, algebraic level and apply its results to each formalism which has a fixpoint semantics.

This approach is especially powerful when combined with the framework of *approximation theory*, a general fixpoint theory for arbitrary operators, which has already proved highly useful in the study of non-monotonic reasoning. It naturally captures, for instance, (most of) the common semantics of logic programming [DMT00], auto-epistemic logic [DMT03] and default logic [DMT03]. As such, studying stratification within this framework, allows our abstract results to be directly and easily applicable to logic programming, auto-epistemic logic and default logic.

Studying stratification at this more *semantical* level has three distinct advantages. First of all, it avoids duplication of effort, as the same algebraic theory takes care of stratification in logic programming, auto-epistemic logic, default logic and indeed any logic with a fixpoint semantics. Secondly, our results can be used to easily extend existing results to other (fixpoint) semantics of the aforementioned languages. Finally, our work also offers greater insight into the general principles underlying various known stratification results, as we are able to study this issue in itself, free of being restricted to a particular syntax or semantics.