1 Introduction

Argumentation is an important cognitive process for dealing with conflicting information by generating and/or comparing arguments. Often it is based on constructing and comparing deductive arguments. These are arguments that involve some premises (which we refer to as the support of the argument) and a conclusion (which we refer to as the claim of the argument) such that the support deductively entails the claim.

In order to formalize argumentation, we could potentially use any logic to define the logical entailment of the claim from the support. Possible logics include defeasible logics, description logics, paraconsistent logics, modal logics, and classical logic. In this chapter, we focus on deductive arguments in the setting of classical logic. Hence, our starting position is that a deductive argument consists of a claim entailed by a collection of statements such that the claim as well as the statements are denoted by formulae of classical logic and entailment is deduction in classical logic. Classical logic is a well-known formalism. It is widely used in philosophy, mathematics, and computer science for capturing deductive reasoning. It has a simple and intuitive syntax and semantics, and it is supported by a proof theory and extensive foundational results. By using classical logic, we can provide a simple and efficient formalization of argument and counterargument.

So in our framework, an argument is simply a pair \( \langle \Phi, \alpha \rangle \) where the first item in the pair is a minimal consistent set of formulae that proves the second item. That is, we account for the support and the claim of an argument though we do not indicate the method of inference since it does not differ from one argument to another: We
only consider deductive arguments, hence the method of inference for each and every argument is always entailment according to classical logic.

A counterargument for an argument \( \langle \Phi, \alpha \rangle \) is an argument \( \langle \Psi, \beta \rangle \) where the claim \( \beta \) contradicts the support \( \Phi \). Furthermore, we identify a particular kind of counterargument called a canonical undercut \( \langle \Psi, \beta \rangle \) where \( \beta \) is equivalent to \( \neg(\phi_1 \land \ldots \land \phi_n) \) and \( \{\phi_1, \ldots, \phi_n\} \) is the support of the argument being undercut. This is a valuable form of undercut since it subsumes many other kinds of undercut, and hence focusing on only canonical undercuts renders the presentation and evaluation of counterarguments as a more manageable process.

Each undercut to an argument is itself an argument, and so may be undercut, and hence by recursion each undercut needs to be considered for its undercuts. Exploring systematically the universe of arguments in order to present an exhaustive synthesis of the relevant chains of undercuts for a given argument is the basic principle of our approach.

Following on from the idea that we can capture undercuts, and by recursion undercuts to undercuts, our notion of an argument tree is that it is a synthesis of all the arguments that challenge the argument at the root of the tree, and it also contains all counterarguments that challenge these arguments and so on recursively. In each instance, the only counterarguments we consider are the canonical undercuts.

In the rest of this chapter, we formalize and illustrate arguments and counterarguments (including canonical undercuts), and show how these can be collected into argument trees. We conclude the chapter with a comparison with other approaches to formalising argumentation. Since the aim of this chapter is to just introduce some of the basic ideas to argumentation based on classical logic, the interested reader is requested to refer to [3, 6] for more details including formal results.

2 Preliminaries

We assume the reader has some knowledge of classical logic. We will represent atoms by lower case roman letters (\( a, b, c, d, \ldots \)), formulae by greek letters (\( \alpha, \beta, \gamma, \ldots \)), and use \( \land, \lor, \rightarrow \), and \( \neg \) to denote the logical connectives conjunction, disjunction, negation, and implication (respectively). We use \( \vdash \) to denote the classical consequence relation, and so if \( \Delta \) is a knowledgebase, and \( \alpha \) is a formula, then \( \Delta \vdash \alpha \) denotes that \( \Delta \) entails \( \alpha \) (or equivalently \( \alpha \) is a consequence of \( \Delta \)). We also use \( \bot \) to denote a contradiction, and so \( \Delta \vdash \bot \) denotes that \( \Delta \) is contradictory (or equivalently inconsistent).

For the knowledgebase, we first assume a fixed \( \Delta \) (a finite set of formulae) and use this \( \Delta \) throughout. So when we consider arguments and counterarguments, they will be formed from this \( \Delta \). For examples, we will explicitly give the elements of the knowledgebase.

We further assume that every subset of \( \Delta \) is given an enumeration \( \langle \alpha_1, \ldots, \alpha_n \rangle \) of its elements, which we call its canonical enumeration. This really is not a demanding constraint: In particular, the constraint is satisfied whenever we impose an arbitrary