Analysis of Dialogical Argumentation via Finite State Machines

Anthony Hunter

Department of Computer Science, University College London,
Gower Street, London WC1E 6BT, UK

Abstract. Dialogical argumentation is an important cognitive activity by which agents exchange arguments and counterarguments as part of some process such as discussion, debate, persuasion and negotiation. Whilst numerous formal systems have been proposed, there is a lack of frameworks for implementing and evaluating these proposals. First-order executable logic has been proposed as a general framework for specifying and analysing dialogical argumentation. In this paper, we investigate how we can implement systems for dialogical argumentation using propositional executable logic. Our approach is to present and evaluate an algorithm that generates a finite state machine that reflects a propositional executable logic specification for a dialogical argumentation together with an initial state. We also consider how the finite state machines can be analysed, with the minimax strategy being used as an illustration of the kinds of empirical analysis that can be undertaken.

1 Introduction

Dialogical argumentation involves agents exchanging arguments in activities such as discussion, debate, persuasion, and negotiation [1]. Dialogue games are now a common approach to characterizing argumentation-based agent dialogues (e.g. [2–12]). Dialogue games are normally made up of a set of communicative acts called moves, and a protocol specifying which moves can be made at each step of the dialogue. In order to compare and evaluate dialogical argumentation systems, we proposed in a previous paper that first-order executable logic could be used as common theoretical framework to specify and analyse dialogical argumentation systems [13].

In this paper, we explore the implementation of dialogical argumentation systems in executable logic. For this, we focus on propositional executable logic as a special case, and investigate how a finite state machine (FSM) can be generated as a representation of the possible dialogues that can emanate from an initial state. The FSM is a useful structure for investigating various properties of the dialogue, including conformance to protocols, and application of strategies. We provide empirical results on generating FSMs for dialogical argumentation, and how they can be analysed using the minimax strategy. We demonstrate through preliminary implementation that it is computationally viable to generate the FSMs and to analyse them. This has wider implications in
using executable logic for applying dialogical argumentation in practical uncertainty management applications, since we can now empirically investigate the performance of the systems in handling inconsistency in data and knowledge.

2 Propositional Executable Logic

In this section, we present a propositional version of the executable logic which we will show is amenable to implementation. This is a simplified version of the framework for first-order executable logic in [13].

We assume a set of atoms which we use to form propositional formulae in the usual way using disjunction, conjunction, and negation connectives. We construct modal formulae using the ⊢, ⊣, ⊕, and ⊖ modal operators. We only allow literals to be in the scope of a modal operator. If α is a literal, then each of ⊕α, ⊖α, ⊢α, and ⊣α is an action unit. Informally, we describe the meaning of action units as follows: ⊕α means that the action by an agent is to add the literal α to its next private state; ⊖α means that the action by an agent is to delete the literal α from its next private state; ⊢α means that the action by an agent is to add the literal α to the next public state; and ⊣α means that the action by an agent is to delete the literal α from the next public state.

We use the action units to form action formulae as follows using the disjunction and conjunction connectives: (1) If φ is an action unit, then φ is an action formula; And (2) If α and β are action formulae, then α ∨ β and α ∧ β are action formulae. Then, we define the action rules as follows: If φ is a classical formula and ψ is an action formula then φ ⇒ ψ is an action rule (which we might use in an example where b denotes belief, and c denotes claim, and a is some information).

Implicit in the definitions for the language is the fact that we can use it as a meta-language [14]. For this, the object-language will be represented by terms in this meta-language. For instance, the object-level formula p(a, b) → q(a, b) can be represented by a term where the object-level literals p(a, b) and q(a, b) are represented by constant symbols, and → is represented by a function symbol. Then we can form the atom belief(p(a, b) → q(a, b)) where belief is a predicate symbol. Note, in general, no special meaning is ascribed the predicate symbols or terms. They are used as in classical logic. Also, the terms and predicates are all ground, and so it is essentially a propositional language.

We use a state-based model of dialogical argumentation with the following definition of an execution state. To simplify the presentation, we restrict consideration in this paper to two agents. An execution represents a finite or infinite sequence of execution states. If the sequence is finite, then t denotes the terminal state, otherwise t = ∞.

Definition 1. An execution e is a tuple e = (s1, a1, p, a2, s2, t), where for each n ∈ N where 0 ≤ n ≤ t, s1(n) is a set of ground literals, a1(n) is a set of ground action units, p(n) is a set of ground literals, a2(n) is a set of ground action units, s2(n) is a set of ground literals, and t ∈ N ∪ {∞}. For each n ∈ N, if 0 ≤ n ≤ t, then an execution state is e(n) = (s1(n), a1(n), p(n), a2(n), s2(n)) where e(0)