On the Complexity of Linking Deductive and Abstract Argument Systems

Michael Wooldridge and **Paul E. Dunne**
Dept of Computer Science
University of Liverpool
Liverpool L69 3BX, UK
mjw,ped@csc.liv.ac.uk

Simon Parsons
Brooklyn College
CUNY
Brooklyn 11210 NY
parsons@sci.brooklyn.cuny.edu

Abstract

We investigate the computational complexity of a number of questions relating to deductive argument systems, in particular the complexity of linking deductive and more abstract argument systems. We start by presenting a simple model of deductive arguments based on propositional logic, and define logical equivalence and defeat over individual arguments. We then extend logical equivalence to sets of arguments, and show that the problem of checking equivalence of argument sets is co-NP-complete. We also show that the problem of checking that an argument set contains no two logically equivalent arguments is NP-complete, while the problem of checking that a set of arguments is maximal (i.e., that no argument could be added without such an argument being logically equivalent to one that is already present) is co-NP-complete. We then show that checking whether a digraph over an argument set is sound with respect to the defeat relation is co-NP-complete, while the problem of showing that such a digraph is complete is NP-complete, and the problem of showing both soundness and completeness is D^p-complete.

Introduction

Argumentation is the process of attempting to construct rationally justifiable set of beliefs (Prakken & Vreeswijk 2001), and is increasingly used as a mechanism to support interaction in multiagent systems (Parsons, Wooldridge, & Amgoud 2003). The argumentation process typically starts with a knowledge base that contains logical conflicts, and is hence inconsistent: argumentation can be understood as the process of extracting a rationally justifiable position from this inconsistent starting point. Essentially two different approaches to formalizing arguments have been put forward in the literature. The first is the *abstract argument* framework of (Dung 1995). In this framework, the starting point is simply a digraph, with vertices in the graph corresponding to arguments, and edges in the graph representing the notion of attack, or defeat, between arguments. Abstract argument systems are so-called because they abstract away from the internal structure and properties of individual arguments, and focus instead simply on the attack relation between arguments. An alternative approach, which instead gives arguments some internal, logical structure, is that of *deductive* argument systems (Pollock 1992; 1994; Krause *et al.* 1995; Amgoud 1999; Besnard & Hunter 2001; Parsons, Wooldridge, & Amgoud 2003).

Our aim in this paper is to investigate the computational complexity of a number of questions relating to deductive argument systems, but in particular, the complexity of *linking* deductive argument systems and abstract argument systems. Such a linking seems to be necessary if we are to use argumentation systems, for example as the basis of inter-agent communication in multiagent systems. To capture the internal structure and intended meaning of arguments, we need something of deductive arguments, while to capture the interactions between arguments, and to formulate appropriate solution concepts, we clearly need something akin to an abstract argument system. Thus, to be practically useful, it seems an argumentation framework must have linked elements of both deductive and abstract argument systems.

We start by presenting a simple model of deductive arguments, and define notions logical equivalence and defeat over individual arguments. We then extend logical equivalence to sets of arguments, and show that the problem of checking equivalence of argument sets is co-NP-complete. We also show that the problem of checking that an argument set is *distinct* (i.e., contains no two logically equivalent arguments) is NP-complete, while the problem of checking that a set of arguments is *maximal* (i.e., that no argument could be added without such an argument being logically equivalent to one that is already present) is co-NP-complete. We then show that checking whether a graph over an argument set is sound with respect to the defeat relation is co-NP-complete, while the problem of showing that such a graph is complete is NP-complete, and the problem of showing both soundness and completeness is D^p-complete.

Deductive Arguments, Defeat, & Equivalence

We present the model of deductive arguments that we work with throughout the remainder of this paper. This model is closely related to those of (Besnard & Hunter 2001; Parsons, Wooldridge, & Amgoud 2003). Let $\Phi_0 = \{p, q, \ldots\}$ be a finite, fixed, non-empty vocabulary of Boolean variables, and let Φ denote the set of (well-formed) formulae of propositional logic over Φ_0, constructed using the conventional Boolean operators ("$\wedge$", "$\vee$", "$\rightarrow$", "$\leftrightarrow$", and "$\neg$"), as well as the truth constants "$\top$" (for truth) and "$\bot$" (for falsity). We refer to a finite subset of Φ as a *database*, and use $\Delta, \Delta', \Delta_0, \ldots$ as variables ranging over the set of Φ-databases. We assume a conventional semantic consequence relation "$\models$" for propositional logic, writing $\Delta \models \varphi$ to mean that φ is a logical consequence of the database Δ. We write $\models \varphi$ as a shorthand for $\emptyset \models \varphi$; thus $\models \varphi$ means that φ is a tautology. We denote the fact that formulae $\varphi, \psi \in \Phi$ are *logically equivalent* by $\varphi \sim \psi$; thus $\varphi \sim \psi$ means that $\models \varphi \leftrightarrow \psi$. Note that "$\sim$" is a *meta-language* relation symbol, which should not be confused with the object-language bi-conditional operator "$\leftrightarrow$".

If $\Delta \subseteq \Phi$ is a database, then an argument, α, over Δ is a pair $\alpha = (C, S)$ where $C \in \Phi$ is a propositional formula which we refer to as the *conclusion* of the argument, and $S \subseteq \Delta$ ($S \neq \emptyset$) is a subset of Δ which we refer to as the *support* of the argument, such that $S \models C$, i.e., C is a logical consequence of S. Notice that we omit two constraints on arguments that are commonly assumed in the literature, namely that S is consistent, and that S is minimal (Besnard & Hunter 2001; Parsons, Wooldridge, & Amgoud 2003). Of the two, minimality is generally regarded as an aesthetic criterion, rather than technically essential. Consistency is more important, and of course by relaxing this constraint we admit into our analysis some scenarios that do not seem to have any useful interpretation; but of course this does not invalidate the results we present. Let $A(\Delta)$ denote the set of arguments over Δ. If α is an argument, then we denote the support of α by $S(\alpha)$ and the conclusion of α by $C(\alpha)$.

There are two common ways of defining defeat between two deductive arguments (Prakken & Vreeswijk 2001): rebuttal (where the conclusion of each argument is logically equivalent to the negation of the conclusion of the other) and undercut (where the conclusion of the attacker contradicts some part of the support of the other). It is not hard to see that the rebuttal relation between arguments will be symmetric, and this potentially limits its value as an analytical concept (Besnard & Hunter 2001). We therefore focus on undercutting (Besnard & Hunter 2001). There are in fact a number of ways of defining undercuts (Besnard & Hunter 2001), and our choice here is largely motivated by simplicity. We say an argument α_1 *defeats* an argument α_2 (written $def(\alpha_1, \alpha_2)$) if $\exists \varphi \in S(\alpha_2)$ such that $C(\alpha_1) \sim \neg\varphi$. The problem of checking whether $def(\alpha_1, \alpha_2)$ is obviously co-NP-complete.

Now, consider the circumstances under which two arguments may be said to be *equivalent*. First, consider the equivalence of formulae: we have two obvious interpretations of "equivalence" w.r.t. formulae. The first is simply that of syntactic equivalence, which we denote by equality ("$=$"); and the second is that of *logical* equivalence, which you will recall is denoted by $\sim$. Let us now extend these notions to arguments. We write $\alpha_1 = \alpha_2$ to mean that α_1 and α_2 are *syntactically equivalent*, i.e., that $C(\alpha_1) = C(\alpha_2)$ and $S(\alpha_1) = S(\alpha_2)$. What about logical equivalence of arguments? We will say that arguments α_1 and α_2 over a database Δ are logically equivalent (written: $\alpha_1 \approx \alpha_2$) iff $C(\alpha_1) \sim C(\alpha_2)$, i.e., if they are in complete logical agreement w.r.t. the conclusion. The point here is that this ignores syntactic variations in the presentation of the argument's conclusion. In general, for any database Δ, there will be more syntactically distinct arguments over Δ than there will be logically distinct arguments over Δ. To see this, simply consider a database $\Delta_1 = \{p \rightarrow q, p\}$, and the arguments $(q, \{p, p \rightarrow q\})$ and $(\neg\neg q, \{p, p \rightarrow q\})$, both of which are syntactically distinct, but $(q, \{p, p \rightarrow q\}) \approx (\neg\neg q, \{p, p \rightarrow q\})$. It is evident that checking whether two arguments are logically equivalent is co-NP-complete.

Argument Sets, Distinctness, & Maximality

We now change our focus to consider subsets of arguments. We extend our notion of equivalence of arguments to subsets of arguments as follows. We say $X_1 \subseteq A(\Delta)$ and $X_2 \subseteq A(\Delta)$ are logically equivalent (written: $X_1 \approxeq X_2$) iff there exists a bijection $f : X_1 \rightarrow X_2$ such that $\forall \alpha \in X_1$, we have $\alpha \approx f(\alpha)$. The $\exists\forall$ pattern of quantifiers in the checking of equivalence of argument sets suggests that this is computationally harder than checking equivalence of formulae or arguments – perhaps Σ_2^p-complete (Papadimitriou 1994, pp.424–425). However, this turns out not to be the case:

Theorem 1 *The problem of checking equivalence of argument sets is co-*NP*-complete.*

Proof: It will be convenient to work with the complementary problem – INEQUIVALENT ARGUMENT SETS (IAS) – and to prove that IAS is NP-complete. Showing NP-hardness is straightforward, so we focus on membership of NP. Let $X_1 = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ and $X_2 = \{\beta_1, \ldots, \beta_m\}$. where $\alpha_i = (\varphi_i, S_i)$ and $\beta_i = (\psi_i, T_i)$ are arguments in $A(\Delta)$. We use Φ and Ψ to denote the sets $\{\varphi_1, \ldots, \varphi_m\}$ and $\{\psi_1, \ldots, \psi_m\}$.

Let $B(\Phi, \Psi, E)$ be the bipartite graph on (disjoint) sets of m vertices labelled Φ and Ψ and whose edges are $E = \{\{\varphi_i, \psi_j\} : S(\alpha_i) = S(\beta_j)\}$. For $\underline{a} \in \langle \bot, \top \rangle^n$, $B_{\underline{a}}(\Phi, \Psi, F_{\underline{a}})$ is the subgraph of $B(\Phi, \Psi, E)$ containing only the edges $F_{\underline{a}} = \{\{\varphi_i, \psi_j\} : \varphi_i(\underline{a}) = \psi_j(\underline{a})\}$. For $Y \subseteq \langle \bot, \top \rangle^n$, $B_Y(\Phi, \Psi, F_Y)$ is the subgraph of $B(\Phi, \Psi, E)$ whose edges are $F_Y = \{\{\varphi_i, \psi_j\} : \varphi_i(\underline{a}) = \psi_j(\underline{a}) \text{ for every } \underline{a} \in Y\}$. Letting TOT denote the set $\langle \bot, \top \rangle^n$ it is easy to see the following: $X_1 \approxeq X_2$ iff $B_{\mathrm{TOT}}(\Phi, \Psi, F_{\mathrm{TOT}})$ contains a perfect matching, i.e., a subset of m edges defining a bijective mapping between Φ and Ψ.

For $V \subset \Phi$, let $\Gamma(V, B_Y)$ denote the subset of Ψ formed by $\Gamma(V, B_Y) = \{\psi_j : \{\varphi_i, \psi_j\} \in F_Y \text{ for some } \varphi_i \in V\}$. From the König-Hall Theorem ((Berge 1976, Ch. 7, Thm. 5, p. 134)), there is a perfect matching in $B_{\mathrm{TOT}}(\Phi, \Psi, F_{\mathrm{TOT}})$ if and only if $\forall\ V \subset \Phi\ |\Gamma(V, B_{\mathrm{TOT}})| \geq |V|$. Suppose it is the case that $X_1 \not\approxeq X_2$, i.e., $\langle X_1, X_2 \rangle$ is accepted as an instance of IAS. From the argument above, this happens if and only if $B_{\mathrm{TOT}}(\Phi, \Psi, F_{\mathrm{TOT}})$ does *not* contain a perfect matching, and thus there will be some strict subset of Φ, V say, for which $|\Gamma(V, B_{\mathrm{TOT}})| < |V|$.

These observations lead to the following NP algorithm to decide IAS.

1. Non-deterministically choose some $V \subset \Phi$.
2. Non-deterministically choose some $W \subset \Psi$ of size $|V| - 1$.

3. Non-deterministically select a set F of $|V|.(m - |W|) < m^2$ distinct $\underline{a} \in$ TOT.
4. For each $\varphi \in V$ and each $\psi \in \Psi \setminus W$ check that if $\{\varphi, \psi\} \in E$ then there is some $\underline{a} \in F$ for which $\psi(\underline{a}) \neq \varphi(\underline{a})$.

The last stage involves only polynomially many tests, each of which requires simply evaluating two formulae on a given instantiation.

To see that this algorithm is correct it suffices to observe that the structure $\langle V, W, F\rangle$ witnesses that $\langle X_1, X_2\rangle$ is a positive instance of IAS: there are at most $|V|.(m - |V|)$ pairs $\langle \varphi_i, \psi_j\rangle$ with $\varphi_i \in V$ and $\psi_j \notin W$. If $S(\alpha_i) = S(\beta_j)$ then $\{\varphi_i, \psi_j\} \in E$, however, we only require one instantiation $\underline{a} \in$ TOT in order to eliminate this edge from F_{TOT}. Thus we need at most $|V|.(m - |V|) < m^2$ instantiations to remove all edges between V and $\Psi \setminus W$. It follows that $\langle V, W, F\rangle$ provides a polynomial length certificate for membership in IAS. □

Next, we define the notion of *distinctness* for sets of arguments. The intuition is that a set of arguments is distinct if it does not contain duplicated arguments, where duplication is measured with respect to logical equivalence of arguments. Formally, we say argument set $X \subseteq A(\Delta)$ is distinct iff $\forall \alpha_1, \alpha_2 \in X$: if $(\alpha_1 \neq \alpha_2)$ then $(\alpha_1 \not\sim \alpha_2)$.

Theorem 2 *The problem of checking whether an argument set is distinct is* NP-*complete.*

Proof: Membership of NP follows from the fact that checking distinctness of an argument set $X \subseteq A(\Delta)$ reduces to the $|X|^2$ independent satisfiability checks, i.e., verifying that for all $\alpha_1, \alpha_2 \in X$, such that $\alpha_1 \neq \alpha_2$, the formula $(C(\alpha_1) \wedge \neg C(\alpha_2)) \vee (\neg C(\alpha_1) \wedge C(\alpha_2))$ is satisfiable. For NP-hardness, we reduce SAT. Given a SAT instance φ, simply check that the argument set $X_1 = \{(\varphi, \{\varphi\}), (\bot, \{\bot\})\}$ is distinct. □

We say a set of arguments $X \subseteq A(\Delta)$ is *maximal* w.r.t. Δ if it is not possible to add an argument from $A(\Delta)$ to X without X becoming indistinct. Intuitively, if a set of arguments X is maximal with respect to some database Δ, then it contains all the arguments that can be made about Δ: it is not possible to pick an argument from $A(\Delta)$ without duplicating a member of X (where duplication is measured with respect to logical equivalence). Notice that distinctness does not of course imply maximality, but *neither does maximality imply distinctness*. That is, an argument set can be maximal but contain duplicates. Indeed, the set $A(\Delta)$ of all arguments that can be made with respect to a database Δ is an obvious example of such a maximal but indistinct set.

In analysing the computational complexity of checking maximality it will be convenient to work with its complementary form, which we dub NON-MAS. In this problem, instances $\langle \Delta, X\rangle$ are accepted iff there is some $\beta \in A(\Delta)$ whose conclusion is *not* logically equivalent to that of any argument in X. We observe that the "natural" formulation of NON-MAS in determining the status of an instance $\langle \Delta, X\rangle$ is as

$$\exists\, S \subseteq \Delta,\ \varphi\ :\ \langle \varphi, S\rangle \in A(\Delta)\ \wedge\ \bigwedge_{\alpha \in X} (\varphi\ \not\sim\ C(\alpha))$$

This formulation raises two difficulties: unless restrictions are placed on φ, the structure $\langle \varphi, S\rangle$ which must be validated as an argument in $A(\Delta)$ may have size that is not polynomially bounded in the size of the instance $\langle \Delta, X\rangle$[1]; we have to validate $S \models \varphi$ (in general CO-NP-hard) and $\varphi \not\sim C(\alpha)$ for each $\alpha \in X$ (in general, NP-hard). Overall, even assuming the restriction to formulae whose size, $|\varphi|$ (measured as the number of literals occurring in φ) is bounded by some polynomial in the instance size, it would appear that NON-MAS is "unlikely" to be decidable by an NP computation: with the formulation and the restriction imposed we get only a Σ_2^p algorithm. However, not only is it unnecessary *explicitly* to restrict φ, we may also validate $S \models \varphi$ for all *relevant* φ in *polynomial time* (in the size of the instance $\langle \Delta, X\rangle$). In this way we can show that NON-MAS $\in$ NP a result which, coupled with the easy proof that NON-MAS is NP-hard, allows us to deduce that NON-MAS is NP-complete.

The following result is central to our subsequent proof that NON-MAS $\in$ NP.

Lemma 1 *Let $\langle \Delta, X\rangle$ be an instance of* NON-MAS *and n be the number of Boolean variables in the vocabulary Φ_0 of Δ. The instance $\langle \Delta, X\rangle$ is accepted if and only if there is a propositional formula, φ, over Φ_0 for which* all *of the following properties hold:*

a. $\Delta \models \varphi$.

b. *φ is a* CNF-*formula containing at most $|X|$ clauses, each of which is defined by* exactly *n literals, so that $|\varphi| \leq n|X|$.*

c. $\forall\ \alpha \in X$, $C(\alpha) \not\sim \varphi$.

Proof: From the definition of NON-MAS it is immediate that if φ with the properties (a) through (c) exists, then it is certainly the case the $\langle \Delta, X\rangle$ is accepted as an instance of NON-MAS: the argument (φ, Δ) being distinguished from all arguments in X.

For the converse implication, suppose it is the case that $(\psi, S) \in A(\Delta)$ and that for each $\alpha \in X$, we have $\psi \not\sim C(\alpha)$. We first observe that, since $S \models \psi$ it is certainly the case that $\Delta \models \psi$. For any instantiation $\underline{a} \in \langle \top, \bot\rangle^n$, let $\chi_{\underline{a}}$ be the propositional formula given as the disjunction over all literals over Φ_0 that take the value $\bot$ under $\underline{a}$: thus $\chi_{\underline{a}}(\underline{b}) = \bot \Leftrightarrow \underline{b} = \underline{a}$. Consider the set of (full) instantiations of Φ_0, $\bot(\psi)$, defined by,

$$\bot(\psi)\ =\ \{\underline{a} \in \langle \top, \bot\rangle^n\ :\ \psi(\underline{a}) = \bot\}$$

It is well-known that for any propositional formula, ψ, is logically equivalent to the formula ψ_{CNF} defined via

$$\psi_{\text{CNF}}\ =\ \bigwedge_{\underline{a} \in \bot(\psi)} \chi_{\underline{a}}$$

Thus from $\Delta \models \psi$ we have $\Delta \models \psi_{\text{CNF}}$. In addition, however, for any subset R of $\bot(\psi)$ it further holds that

$$\Delta \models\ \left(\bigwedge_{\underline{a} \in R} \chi_{\underline{a}}\right)$$

[1]Given a database Δ, it may be the case that there is some φ such that $\Delta \models \varphi$ and the shortest formula ψ such that $\varphi \sim \psi$ is of length exponential in the size of Δ. In other words, there could be arguments that we can construct from a database whose conclusion is necessarily exponential in the size of the database.

Since $\psi_{\text{CNF}} \not\sim C(\alpha)$ for any $\alpha \in X$, it follows that we can identify $k = |X|$ instantiations, $\langle \underline{a}_1, \underline{a}_2, \ldots, \underline{a}_k \rangle$ for which $\psi_{\text{CNF}}(\underline{a}_i) \neq C(\alpha_i)(\underline{a}_i)$. We now define the subset R_X of $\bot(\psi)$ to contain $\{ \underline{a}_i : C(\alpha_i)(\underline{a}_i) = \top \}$ and fix φ (the propositional formula whose existence we wish to establish) as $\bigwedge_{\underline{a} \in R_X} \chi_{\underline{a}}$. For φ defined in this way, from our earlier analysis: $\Delta \models \varphi$, as required by (a); φ is in CNF, and contains at most $|X|$ clauses (since $|R_X| \leq |X|$) with each clause defined from exactly n literals – as required by (b); finally $\varphi \not\sim C(\alpha)$ for any $\alpha \in X$ – as required by (c). To see that (c) does hold true of φ it suffices to observe that $\bot(\varphi) = R_X$ so that if $\underline{a}_i \in R_X$ then $\varphi(\underline{a}_i) = \bot$ and (from the definition of R_X) $C(\alpha_i)(\underline{a}_i) = \top$; similarly if $\underline{a}_i \notin R_X$ then $C(\alpha_i)(\underline{a}_i) = \bot$ and (from the fact that $\bot(\varphi) = R_X$) $\varphi(\underline{a}_i) = \top$.

In total if it is the case that $\langle \Delta, X \rangle$ is accepted as an instance of NON-MAS, then we can identify some φ with the properties (a)–(c) described in the Lemma statement. □

Given this, we can now prove that:

Theorem 3 *The problem of checking maximality of argument sets is* CO-NP-*complete.*

Proof: We prove the equivalent result that NON-MAS is NP-complete. We first show NON-MAS is NP-hard using a reduction from SAT. Given an instance φ of SAT, consider the database $\Delta = \{\neg\varphi\}$ with $X \subseteq A(\Delta)$ chosen to be $\{(\top, \{\neg\varphi\})\}$. We claim that φ is satisfiable iff X is not maximal w.r.t. Δ.

We now show that NON-MAS $\in$ NP. Consider the following non-deterministic algorithm.

1. For each $\alpha_i \in X$, non-deterministically choose an instantiation, $\underline{a}_i$ of Φ_0.
2. Construct the formula $\varphi = \bigwedge_{\underline{a}_i : C(\alpha_i)(\underline{a}_i) = \top} \chi_{\underline{a}_i}$
3. Test if $\Delta \models \varphi$, accepting if this is the case.

By Lemma 1 it is certainly the case that $\langle \Delta, X \rangle$ is accepted as an instance of NON-MAS if and only if the algorithm described has an accepting computation. Stages (1) and (2) can clearly be realised in non-determininistic polynomial time. The final stage, however, is easily completed in deterministic polynomial-time: φ is a CNF formula for which $\bot(\varphi) = \{\underline{a}_i : C(\alpha_i)(\underline{a}_i) = \top\}$. Thus to verify $\Delta \models \varphi$ it suffices to check that for each $\underline{a} \in \bot(\varphi)$ some $\psi \in \Delta$ has $\psi(\underline{a}) = \bot$. Since $|\bot(\varphi)| \leq |X|$ this final stage takes time polynomial in the size of the instance $\langle \Delta, X \rangle$. □

Of particular interest to us are argument sets over Δ that are *both* maximal *and* distinct. We say a set of arguments X is *canonical* with respect to Δ if it is both maximal and distinct w.r.t. Δ. A canonical argument set thus represents a limit of what can be argued from a database without repetition. We will let $can(\Delta)$ denote the canonical argument sets of Δ, so $can(\Delta) \subseteq 2^{A(\Delta)}$. First, we prove that every non-empty database Δ has a canonical argument set.

Theorem 4 *For all $\Delta \neq \emptyset \subseteq \Phi$, $can(\Delta) \neq \emptyset$.*

Proof: We use the same proof idea as Lindenbaum's lemma. Let $\sigma : \alpha_0, \alpha_1, \ldots$ be an enumeration of arguments over Δ: such an enumeration clearly exists. Corresponding to σ, define a sequence of argument sets $X_0, X_1, \ldots$ where $X_0 = \{\alpha_0\}$, and for $n > 0$,

$$X_n = \begin{cases} X_{n-1} \cup \{\alpha_n\} & \text{if } X_{n-1} \cup \{\alpha_n\} \text{ is distinct} \\ X_{n-1} & \text{otherwise.} \end{cases}$$

Finally, define an argument set X by: $X = \bigcup_{n=0}^{\infty} X_n$. By construction, X will be a canonical argument set of Δ. □

The following, easily established result gives our motivation for using the term "canonical".

Fact 1 *Canonical argument sets are logically equivalent. That is, $\forall X_1, X_2 \in can(\Delta)$: $X_1 \approx X_2$.*

We note, in addition, the following consequence of Lemma 1, the proof of which is omitted.

Corollary 1 *If $X \in can(\Delta)$, then $|X| = 2^{|\bot(\delta)|}$, where $\delta = \bigwedge_{\varphi \in \Delta} \varphi$.*

Argument Graphs

Let us now consider the issue of linking deductive and abstract argument systems. Given a set of arguments $X \subseteq A(\Delta)$, the defeat predicate $def(\cdots)$ induces a graph $(X, \{(\alpha_1, \alpha_2) \mid \alpha_1, \alpha_2 \in X, def(\alpha_1, \alpha_2)\})$ over X, which can obviously be understood as being analogous to the graph structures of Dung's abstract argument systems (Dung 1995). Note that there are some technical difficulties involved in "lifting" a defeat relation to a Dung argumentation graph in this way. In particular, Besnard and Hunter show that unless modified, Dung's notion of an admissible set turns out to collapse under this interpretation; although the notion of an admissible set can be refined to make more sense when interpreted for deductive argument systems, this comes at the cost of eliminating some apparently reasonable cases (Besnard & Hunter 2001). Thus, solution concepts which make sense when studied with respect to arbitrary graphs do not necessarily make sense when the defeat relation is given a concrete interpretation in terms of deductive arguments, suggesting a need for refined versions of these. However, the issue of formulating appropriate Dung-style solution concepts for deductive argument systems is somewhat tangential to the paper at hand, and we shall not investigate this particular issue. Instead, we focus on the problems of establishing links between deductive and more abstract argument systems.

To motivate the discussion, suppose we are given a set of arguments $X \subseteq A(\Delta)$ (for some Δ), and a graph $G_X = (X, E \subseteq X \times X)$, so that the vertices of G_X are the members of X. How might X and G_X be related? Two obvious questions then suggest themselves:

1. *Soundness*: Does G_X "correctly" represents the defeat relation $def(\cdots)$ over X? Formally, G_X will be sound with respect to X iff $\forall \alpha_1, \alpha_2$, if $G_X(\alpha_1, \alpha_2)$ then $def(\alpha_1, \alpha_2)$.
2. *Completeness*: Does G_X "completely" represents the defeat relation $def(\cdots)$ over X? Formally, G_X will be complete with respect to X iff $\forall \alpha_1, \alpha_2$, if $def(\alpha_1, \alpha_2)$ then $G_X(\alpha_1, \alpha_2)$.

Theorem 5 *Given a set of arguments X, the problem of checking whether a graph $G_X = (X, E \subseteq X \times X)$ is sound with respect to the defeat relation $def(\cdots)$ over X is co-*NP*-complete.*

Proof: Consider membership of co-NP. Recall that soundness of G_X with respect to X means that $\forall \alpha_i, \alpha_j \in X$, if $G_X(\alpha_i, \alpha_j)$ then $def(\alpha_i, \alpha_j)$. We work with the complement of the problem, i.e., the problem of showing that $\exists \alpha_i, \alpha_j \in X$ such that $G_X(\alpha_i, \alpha_j)$ and not $def(\alpha_i, \alpha_j)$. The following NP algorithm decides the problem: (i) Guess $\alpha_i, \alpha_j \in X$ and k propositional valuations $\xi_1, \ldots, \xi_k$, where $k = |S(\alpha_j)|$; (ii) Verify that $G_X(\alpha_i, \alpha_j)$ and that $\xi_1 \models (C(\alpha_i) \wedge \psi_1) \vee \neg(C(\alpha_i) \vee \psi_1), \xi_2 \models (C(\alpha_i) \wedge \psi_2) \vee \neg(C(\alpha_i) \vee \psi_2), \ldots, \xi_k \models (C(\alpha_i) \wedge \psi_k) \vee \neg(C(\alpha_i) \vee \psi_k)$, where $S(\alpha_j) = \{\psi_1, \ldots, \psi_k\}$. The algorithm is clearly in NP. For hardness, we reduce TAUT, the problem of deciding whether a propositional logic formula φ is a tautology. Given a TAUT instance φ, define $G_X = (\{\alpha_1, \alpha_2\}, \{(\alpha_1, \alpha_2)\})$ where $\alpha_1 = (\varphi, \{\varphi\})$ and $\alpha_2 = (\perp, \{\perp\})$. G_X is sound w.r.t. X iff φ is a tautology. $\square$

Theorem 6 *Given a set of arguments X, the problem of checking whether a graph $G_X = (X, E \subseteq X \times X)$ is complete with respect to the defeat relation $def(\cdots)$ over X is* NP*-complete.*

Proof: Membership of NP is by the following algorithm: For each $\alpha_1, \alpha_2 \in X$ such that not $G_X(\alpha_1, \alpha_2)$, and for each $\varphi \in S(\alpha_2)$ guess a valuation ξ and verify that $\xi \models C(\alpha_1) \wedge \varphi$. For NP-hardness we reduce SAT. Given a SAT instance φ, define $G_X = (\{\alpha_1, \alpha_2\}, \{(\alpha_1, \alpha_1), (\alpha_2, \alpha_2), (\alpha_2, \alpha_1)\})$ where $\alpha_1 = (\varphi, \{\varphi\})$ and $\alpha_2 = (\top, \{\top\})$. G_X is complete w.r.t. X iff φ is satisfiable. $\square$

Now consider the problem of determining whether a graph G_X over a set of arguments X is both sound *and* complete.

Theorem 7 *Given a set of arguments X, the problem of checking whether a graph $G_X = (X, E \subseteq X \times X)$ is both sound and complete with respect to the defeat relation $def(\cdots)$ over X is D^p-complete.*

Proof: Membership in D^p follows from Theorems 6 and Theorem 5. For completeness, we reduce SAT-UNSAT (Papadimitriou 1994, p.415), instances of which comprise a pair $\langle \varphi, \psi \rangle$ of propositional formulae. Such an instance is accepted if φ is satisfiable *and* ψ is unsatisfiable. First, from φ we create a new formula $\varphi^* = \varphi \wedge p$, where p is a new Boolean variable, which does not appear in either φ or ψ. The formula φ^* has the following properties, all of which are used in what follows: (i) φ^* will be satisfiable iff φ is satisfiable; (ii) φ^* is *not* a tautology, even if φ is; and (iii) neither $\varphi^* \sim \psi$ nor $\varphi^* \sim \neg\psi$. We then create an argument set $X = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ where: $\alpha_1 = (\varphi^*, \{\varphi^*\})$, $\alpha_2 = (\psi, \{\psi\})$, $\alpha_3 = (\top, \{\top\})$, and $\alpha_4 = (\perp, \{\perp\})$. Our argument graph $G_X = (X, E)$ has X as defined above, and $E = \{(\alpha_2, \alpha_3), (\alpha_2, \alpha_2), (\alpha_3, \alpha_4), (\alpha_4, \alpha_3)\}$. Table 1 relates the graph G_X in this construction to the defeat relation $def(\cdots)$ induced over X for possible properties of φ and ψ. Note that the properties in the $def(\cdots)$ column of Table 1 are by established by simple propositional logic reasoning.

(α_i, α_j)	$def(\alpha_i, \alpha_j)$?	$G_X(\alpha_i, \alpha_j)$?
(α_1, α_1)	no	no
(α_1, α_2)	no	no
(α_1, α_3)	iff φ^* is unsatisfiable	no
(α_1, α_4)	no (since φ^* is not a tautology)	no
(α_2, α_1)	no	no
(α_2, α_2)	no	no
(α_2, α_3)	iff ψ is unsatisfiable	yes
(α_2, α_4)	iff ψ is a tautology	no
(α_3, α_1)	iff φ^* is unsatisfiable	no
(α_3, α_2)	iff ψ is unsatisfiable	yes
(α_3, α_3)	no	no
(α_3, α_4)	yes	yes
(α_4, α_1)	no (since φ^* is not a tautology)	no
(α_4, α_2)	iff ψ is a tautology	no
(α_4, α_3)	yes	yes
(α_4, α_4)	no	no

Table 1: Defeat relation and argument graph properties for the construction of Theorem 7.

We claim that φ is satisfiable and ψ is unsatisfiable iff G_X is sound and complete w.r.t. X. ($\rightarrow$) Suppose φ is satisfiable and ψ is unsatisfiable. We must show that G_X is sound and complete w.r.t. X. Soundness means that if $G_X(\alpha_i, \alpha_j)$ then $def(\alpha_i, \alpha_j)$. With respect to Table 1, this means showing that a "yes" in the $G_X(\alpha_i, \alpha_j)$ column implies a "yes" in the $def(\alpha_i, \alpha_j)$ column. That $def(\alpha_3, \alpha_4)$ and $def(\alpha_4, \alpha_3)$ is obvious, so consider whether $def(\alpha_2, \alpha_3)$: since by assumption ψ is unsatisfiable, then it must defeat α_3. Completeness means that if not $G_X(\alpha_i, \alpha_j)$ then not $G_X(\alpha_i, \alpha_j)$. This can be verified by examination of Table 1. ($\leftarrow$) Suppose G_X is sound and complete w.r.t. X, i.e., that $G_X(\alpha_i, \alpha_j)$ iff $def(\alpha_i, \alpha_j)$. We must show that this implies φ is satisfiable and ψ is unsatisfiable. This can be done by examination of cases in Table 1. $\square$

Suppose that instead of being given a graph over a set of arguments, we are given an *arbitrary* graph, $G = (V, E)$, where V is simply an abstract set of vertices and $E \subseteq V \times V$, and we are asked whether G "captures" a given deductive argument system. Here, G really is simply a Dung-style argument system: the nodes in the graph are not arguments, and hence we are not given any interpretation of them with respect to the given deductive argument system. How might we establish a link between such a graph and a deductive argument system? It depends on the way in which the deductive argument system itself is presented:

1. as a graph G_X over $A(\Delta)$;
2. as a (sub)set of arguments $X \subseteq A(\Delta)$; or
3. as a database $\Delta \subseteq \Phi$.

In the first case, we are given *both* a graph $G = (V, E)$ and an argument graph $G_X = (X, E_X \subseteq X \times X)$. The problems of soundness and completeness in this case reduce to standard graph theoretic problems: Soundness means checking whether G is isomorphic to some subgraph of G_X, while completeness means checking whether G_X is isomorphic to some subgraph of G, and checking soundness *and* completeness means checking that G and G_X are isomorphic. From standard results in complexity theory, (in particular the fact that the SUBGRAPH ISOMORPHISM problem is NP-complete) it follows immediately that the problems of checking soundness or completeness for this representation are both NP-complete. The problem of checking both soundness *and* completeness, however, is exactly the well known open problem GRAPH ISOMORPHISM. A classification of the complexity of this problem would in itself represent a major event in the theory of computational complexity.

With respect to the second representation, we are given a set of arguments $X \subseteq A(\Delta)$ and a graph $G = (V, E)$. Here, we have less information: we have no argument graph to compare G with, just a set of arguments X. Thus we do not know a priori what the vertices of G are supposed to correspond to in X – we thus need to "interpret" vertices in G with respect to members of X in our definitions of soundness and completeness. Formally, we will say:

- a graph $G = (V, E)$ is a *sound abstraction* of a set of arguments $X \subseteq A(\Delta)$ if there exists an injective function $f : V \to X$ such that $\forall v_1, v_2 \in V$, if $G(v_1, v_2)$ then $def(f(v_1), f(v_2))$; and
- a graph $G = (V, e)$ is a *complete abstraction* of $X \subseteq A(\Delta)$ iff there exists an injective function $f : X \to V$ such that $\forall \alpha_1, \alpha_2 \in X$, if $def(\alpha_1, \alpha_2)$ then $G(f(\alpha_1), f(\alpha_2))$.

The proofs of Theorems 5 and 6 can be readily adapted to show the following:

Theorem 8 *The problem of checking whether a graph G is a sound abstraction of a set of arguments $X \subseteq A(\Delta)$, is co-*NP*-hard, while the problem of checking whether a graph G is a complete abstraction of a set of arguments $X \subseteq A(\Delta)$, is* NP*-complete.*

With respect to the third representation, we are given simply a database Δ and a graph $G = (V, E)$. This case seems the most elaborate computationally but also perhaps the least interesting practically. Once again, we are given even less information to work with: we only have the database of formulae from which arguments may be constructed. So, how are we to interpret the soundness and completeness questions? Recalling that for any database $\Delta \subseteq \Phi$, the set of canonical argument sets over Δ is denoted by $can(\Delta)$, we can give the following interpretation to soundness and completeness for graphs $G = (V, E)$ against databases Δ:

- a graph $G = (V, E)$ is a *sound canonical abstraction* of a database Δ if $\exists X \in can(\Delta)$ such that G is a sound abstraction of X; and
- a graph $G = (V, E)$ is a *complete canonical abstraction* of a database Δ if $\exists X \in can(\Delta)$ such that G is a complete abstraction of X.

It should be clear that these concepts, are much more baroque (and much less amenable to formal analysis) than those we have studied above. They involve quantifying over canonical argument sets, which as we noted in Corollary 1 will be exponentially large in the number of falsifying assignments for Δ, and hence in general doubly exponential in the number of Boolean variables. We will thus not investigate these latter problems further here.

Related Work & Conclusions

The work described in this paper has been concerned with the computational complexity of answering certain questions about sets of arguments. The particular questions we have considered have not previously been considered, but there are several authors whose work is related to ours in one way or another. For example, the work of Besnard and Hunter (Besnard & Hunter 2001) has some elements in common with our work — a definition of equivalence between arguments that is the same as ours, and a notion of canonicity of argument. Their work, however, is focussed exclusively on the properties of a specific deductive argumentation system while ours deals with properties that apply to a range of argumentation systems. Other authors have considered the computational complexity of answering questions related to arguments. Most notable, perhaps, is the work of Dimopoulos *et al.* who have investigated the complexity of computing the acceptability of individual deductive arguments — a surprisingly hard process because of the recursive nature of the relationships between arguments (Dimopoulos, Nebel, & Toni 2002).

References

Amgoud, L. 1999. *Contribution a l'integration des préferences dans le raisonnement argumentatif.* Ph.D. Dissertation, l'Université Paul Sabatier, Toulouse. (In French).

Berge, C. 1976. *Graphs and Hypergraphs.* North-Holland.

Besnard, P., and Hunter, A. 2001. A logic-based theory of deductive arguments. *Artificial Intelligence* 128:203–235.

Dimopoulos, Y.; Nebel, B.; and Toni, F. 2002. On the computational complexity of assumption-based argumentation for default reasoning. *Artificial Intelligence* 141(1):57–78.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and *n*-person games. *Artificial Intelligence* 77:321–357.

Krause, P.; Ambler, S.; Elvang-Gøransson, M.; and Fox, J. 1995. A logic of argumentation for reasoning under uncertainty. *Computational Intelligence* 11:113–131.

Papadimitriou, C. H. 1994. *Computational Complexity.* Addison-Wesley: Reading, MA.

Parsons, S.; Wooldridge, M.; and Amgoud, L. 2003. Properties and complexity of some formal inter-agent dialogues. *Journal of Logic and Computation* 13(3):347–376.

Pollock, J. L. 1992. How to reason defeasibly. *Artificial Intelligence* 57:1–42.

Pollock, J. L. 1994. Justification and defeat. *Artificial Intelligence* 67:377–407.

Prakken, H., and Vreeswijk, G. 2001. Logics for defeasible argumentation. In Gabbay, D., and Guenther, F., eds., *Handbook of Philosophical Logic (second edition).* Kluwer Academic Publishers: Dordrecht, The Netherlands.