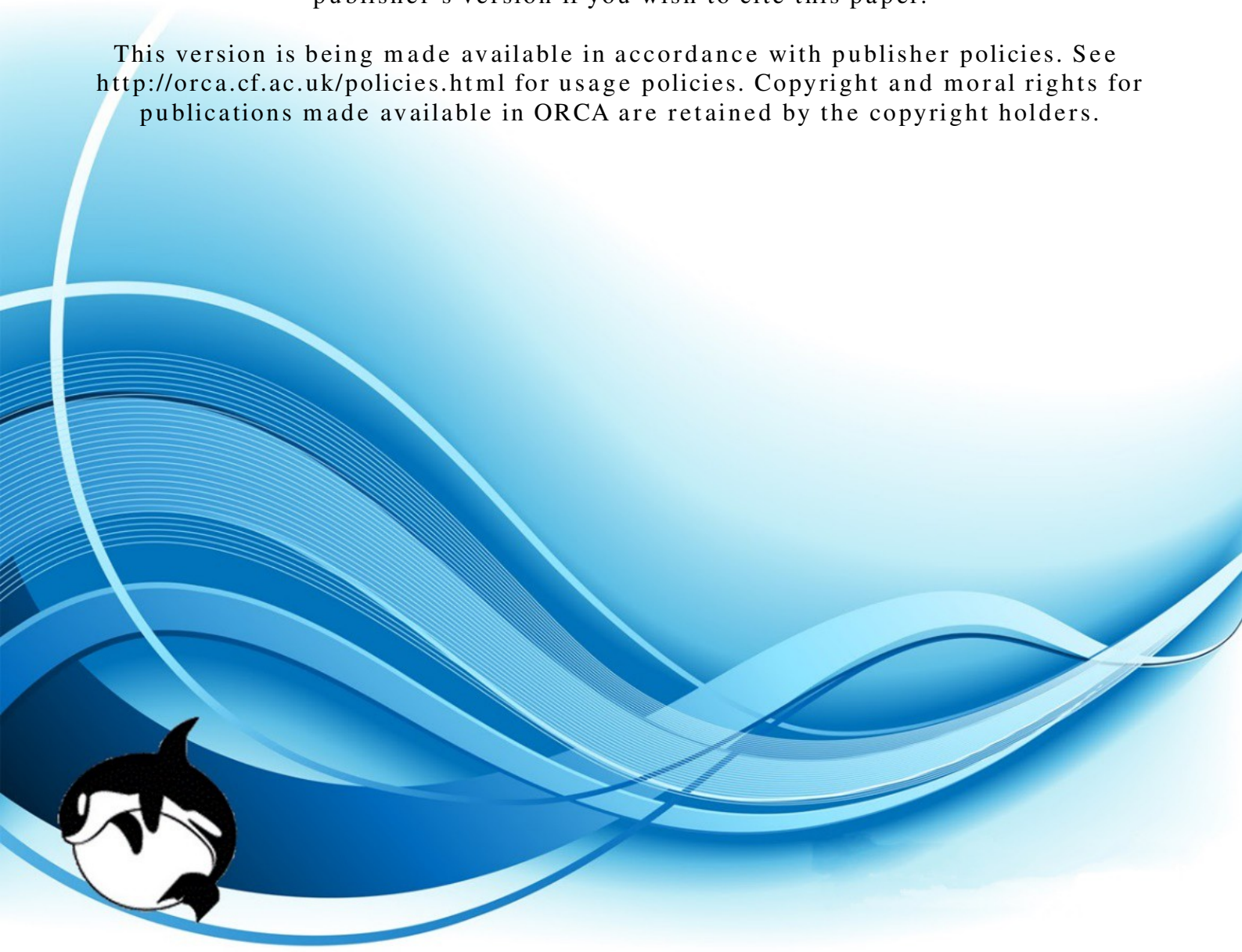

Automata for Infinite Argumentation Structures

Pietro Baroni[a], Federico Cerutti[a,*], Paul E. Dunne[b], Massimiliano Giacomin[a]

[a] *Dipartimento di Ingegneria dell'Informazione, University of Brescia, via Branze, 38, 25123, Brescia, Italy*
[b] *Department of Computer Science, Ashton Building, University of Liverpool, Liverpool L69 7ZF, United Kingdom*

Abstract

The theory of abstract argumentation frameworks (AFs) has, in the main, focused on finite structures, though there are many significant contexts where argumentation can be regarded as a process involving *infinite* objects. To address this limitation, in this paper we propose a novel approach for describing infinite AFs using tools from formal language theory. In particular, the possibly infinite set of arguments is specified through the language recognized by a deterministic finite automaton while a suitable formalism, called *attack expression*, is introduced to describe the relation of attack between arguments. The proposed approach is shown to satisfy some desirable properties which can not be achieved through other "naive" uses of formal languages. In particular, the approach is shown to be expressive enough to capture (besides any arbitrary finite structure) a large variety of infinite AFs including two major examples from previous literature and two sample cases from the domains of multi-agent negotiation and ambient intelligence. On the computational side, we show that several decision and construction problems which are known to be polynomial time solvable in finite AFs are decidable in the context of the proposed formalism and we provide the relevant algorithms. Moreover we obtain additional results concerning the case of *finitary* AFs.

Keywords: Infinite argumentation frameworks, Automata-based representation, Argumentation semantics computation

1. Introduction

The theory of abstract argumentation frameworks (AFs) has advanced considerably since its original formulation in the work of Dung [37]. Now recognised as a core research topic within the field of AI in general and its sub-disciplines

*Corresponding author
Email addresses: pietro.baroni@ing.unibs.it (Pietro Baroni), federico.cerutti@ing.unibs.it (Federico Cerutti), P.E.Dunne@liverpool.ac.uk (Paul E. Dunne), massimiliano.giacomin@ing.unibs.it (Massimiliano Giacomin)

concerned with knowledge representation and multiagent systems in particular, AFs have proven a powerful modelling tool to address reasoning issues in contexts where classical deductive logic is not the most suitable technique. An overview of the role of argumentation in AI may be found in recent surveys such as that of Bench-Capon and Dunne [14] or the comprehensive collection of introductory articles in Rahwan and Simari [77]. In total matters of semantics, algorithms and computational complexity have occupied many researchers to the extent that their key properties are, now, reasonably well understood. Partly in consequence of such understanding, a rich body of subsequent work has emerged promoting developments of Dung's basic formalism in order to encompass scenarios within which the purely abstract approach of [37] is felt to be too limiting. Among the many notable contributions of this nature one finds proposals such as the preference-based AFs of Amgoud and Cayrol [2]; the value-based model of Bench-Capon [15]; EAFs from Modgil [64]; recursive attacks in the AFRA mechanism from Baroni *et al.* [5]; divers treatments of weighted frameworks such as Dunne *et al.* [43]; as well as sophisticated developments of the basic "binary attack" concept from [37] such as the ADF model from Brewka and Woltran [25, 24] and the constrained AFs of Coste-Marquis *et al.* [32, 35].

Amidst the wealth and variety of treatments stemming from [37] one can, however, note a common factor: invariably discussion is focused on *finite* environments be they finite sets of basic arguments or finite attack relationships over these. In contrast, consideration of *infinite* scenarios has been largely neglected. This turns out to be a limitation from a theoretical, conceptual, and practical perspective.

From a theoretical viewpoint, infinite frameworks extend (and, in a sense, complete) the range of investigation on abstract argumentation semantics and their properties. In fact, infinite AFs have been the subject of specific attention in the seminal paper by Dung, whose fundamental results do not rely on finiteness. Subsequently, infinite AFs have sometimes been considered *per se* as significant testbeds for examining semantics properties which, though holding in the finite case, may be challenging to prove or fail to hold in the infinite case. For instance, the existence of semi-stable extensions [29] is guaranteed for finite frameworks, while an infinite framework admitting no semi-stable extensions has been devised in [30] (and will be recalled in Section 7.1) and the existence of semi-stable extensions for *finitary* frameworks has been proved in [91]. Apart from theoretical interest, this kind of results may be useful to shed new light on fundamental issues underlying the definition of different semantics, thus enabling a broader view and deeper understanding in comparing and assessing them.

From a conceptual perspective, considering finite frameworks corresponds to (i) adopting a closed view of the argumentation process, which is bounded to terminate after a finite number of steps and (ii) excluding reasoning about infinite domains. Assumption (i) contrasts with the intrinsically open nature of the argumentation process, arising from the fundamental distinction between the concepts of "demonstration by *proof*" and "persuasion through *argument*". That is to say, as noted in [14, p. 620]: "Arguments are *defeasible*: the reasoning

that formed a persuasive case, in the light of changes of viewpoint or awareness of information not previously available, may fail to convince. This defeasibility is never removed: an argument may cease to be challenged and so accepted, but the *possibility* of challenge remains.". In other words, the finite view can (at best) describe a "snapshot" of the notional complete context within which an argumentation process could evolve due to potential (but as yet unvoiced) challenge to the conclusions derived. On the other hand, limitation (ii) prevents the use of the abstract argumentation formalism in contexts where reasoning has to deal with open-ended scenarios. For instance, considering an open time horizon, one may want to encompass the existence of infinite arguments associated with infinitely many time instants: this kind of approach has been proposed by Pollock to model reasoning about temporal persistence of beliefs [74].

From the practical perspective, it has to be remarked that systems giving rise to a potentially infinite automated production of arguments may well occur in actual applications. On one hand, there are well-known correspondences between argumentation frameworks and other kinds of reasoning systems potentially producing infinite derivations: an example concerning logic programming is in fact given in Appendix A of [37] and will be recalled in this paper. On the other hand, abstract argumentation is widely adopted as a general tool to model dialogues with different purposes (e.g. deliberation, negotiation, persuasion) between self-interested agents in a multi-agent system. In such a context the opportunistic behavior of each agent, driven by "selfish" criteria, and the absence of global coordination and shared information may lead to non-terminating argument exchanges [10].

Adding to the considerations above the fact that, as already recalled, Dung's original work not only addresses infinite AFs as objects of interest but also establishes a number of fundamental properties of such in the context of the basic semantics put forward for AFs, it appears that the very limited coverage of infinite frameworks in the subsequent literature represents an important and, to some extent, surprising lacuna in the field.

This work contributes to fill this gap by addressing the problem of defining finite specifications of infinite AFs through formal languages and proposing an approach based on finite automata. This proposal consists of two basic elements: a description of the (infinite) set of arguments through a finite automaton, and a description of the attack relations linking arguments together through an *attack expression*. More precisely, the attack expression specifies a mapping between regular expressions describing sets of arguments, with the intended meaning that a set S is attacked by the elements of the set obtained from S through the mapping. The combination of the automaton describing the set of arguments and of the attack expression will be called the AF *specification* and will be shown to be expressive enough to encompass a variety of infinite argumentation frameworks, including the major examples available in the literature.

Clearly, a sufficiently expressive specification formalism needs to be complemented by some suitable computational mechanism for the evaluation of argument justification status, which represents the main goal of any application of computational argumentation, either in finite or infinite contexts. The

proposed approach is shown to be satisfactory also from this viewpoint, being able to support the definition of suitable algorithms for some "standard" computational problems in argumentation semantics.

The paper is organised as follows. We lay down the context of the work, discuss motivations, and introduce application examples in Section 2. We then recall the necessary technical background on Dung's argumentation framework and on formal languages in Section 3. In Section 4 we state a set of basic requirements for a representation formalism for infinite frameworks and show that they are not satisfied by a straightforward *naive* approach one may adopt. Section 5 introduces and illustrates with a trailing set of examples the AF *specification* formalism for infinite frameworks and shows that it is expressive enough to capture also "regular" finite frameworks and frameworks that can be regarded as composed by a finite subframework and one or more infinite subframeworks. Section 6 introduces effective computational procedures[1] for several standard decision problems in argumentation semantics in the context of the AF specification formalism. Section 7 demonstrates the suitability of the approach, both on the representation and on the computational side, by analyzing in detail its application to four examples: two infinite frameworks previously introduced in the literature for the sake of theoretical analysis and two examples of infinite argumentation in multi-agent systems taken from Section 2. Section 8 discusses related works whilst Section 9 concludes and proposes some areas for future work. Appendix A recalls the basics of formal language and automata theory to make the paper self-contained, while proofs of all technical results are collected in Appendix B.

2. Context and motivations

Quoting Prakken [76], Dung's paper on abstract argumentation framework "was a breakthrough in three ways: it provided a general and intuitive semantics for the consequence notions of argumentation logics (and for nonmonotonic logics in general); it made a precise comparison possible between different systems (by translating them into his abstract format); and it made a general study of formal properties of systems possible, which are inherited by instantiations of his framework." Due to its abstract nature, Dung's formalism "is best seen not as a formalism for directly representing argumentation-based inference problems but as a tool for analysing particular argumentation systems and for developing a metatheory of such systems". In this perspective, investigation of infinite AFs finds its motivations in the variety of more concrete contexts and systems where infinite structures play a role and that can be modeled using AFs. A (non exhaustive) account is given in the following subsections.

[1] According to [80, p. 55] (but also [63, p. 210] and others) an *effective computational procedure* is "a method each step of which is precisely predetermined and which is certain to produce the answer in a finite number of steps".

2.1. Argumentation models and systems

Infinite entities have been encompassed at a foundational level by all the main literature approaches to formal argumentation. In fact, both before and after Dung's work, other "less abstract" approaches have been defined which formalize arguments and their structure in various ways, with the common property of referring to a generic (and often only partially specified) language, encompassing infinite structures.

In the formalization of defeasible reasoning by Simari and Loui [81] an *argument structure* comprises a possibly infinite set of sentences supporting a conclusion, and, in turn, the set of all possible argument structures is infinite.

In the theory of *assumption-based argumentation* [23] an assumption-based framework consists of a set of beliefs and a set of assumptions which are possibly infinite subsets of a language consisting of countably many sentences. Again the infinite case is explicitly considered in the theoretical analysis of semantics properties in the framework.

In Vreeswijk's *abstract argumentation systems* [89, 9] arguments are restricted to have a finite set of premises, but the set of arguments is possibly infinite and infinite argumentation sequences (involving either finite or infinite sets of arguments) are used as a formal tool for extension evaluation, by introducing a notion of "limit" for infinite argumentation sequences. In [89] it is remarked in particular that some desirable limit properties of infinite argumentation sequences may not hold when an infinite set of arguments is considered. This is left as an open problem, which, as to our knowledge, has still to be solved.

In DEFLOG [87] the central notion is the "dialectical interpretation" of a theory, which basically is a (possibly infinite) set of sentences related by two connectives representing support and defeat relations respectively.

Finally, in the more recent ASPIC formalism [1] and its ASPIC+ evolution [76] arguments with an infinite number of subarguments (and hence infinite sets of arguments) are encompassed and none of the main properties relies on argument finiteness.

It can hence be stated that the consideration of infinite structures and derivations has been consistently regarded as a basic, one could even say "natural", feature in argumentation formalisms. It has however to be acknowledged that this feature has typically been regarded as problematic when moving from (more or less abstract) theory to complete specification (and implementation) of actual argumentation-based reasoning systems. In fact, the unbounded open nature of argumentative reasoning has often been contrasted with the practical needs and limitations of resource-bounded agents.

From a philosophical stance, this contrast has been pointed out by Pollock [72] by introducing the distinction between *justified beliefs* and *warranted propositions*. Quoting Pollock, "at each stage of reasoning, if the reasoning is correct then a belief held on the basis of that reasoning is justified, even if subsequent reasoning will mandate its retraction" while "in contrast to justification, warrant is what the system of reasoning is ultimately striving for. A proposition is

warranted in a particular epistemic situation if and only if, starting from that epistemic situation, an ideal reasoner unconstrained by time or resource limitations would ultimately be led to believe the proposition". In this view, warrant may be regarded as a sort of unattainable goal for a pratical resource-bounded agent, which needs to be content with the more limited notion of justification.

From the practical side, an example of the problems in dealing with infinite structures is presented in [52] in the context of the *DeLP (Defeasible Logic Programming)* system. Here, one of the central notions is the one of an "argumentation line", which is basically a sequence of argument structures where each element of the sequence is a defeater of its predecessor. As discussed in [52] infinite argumentation lines may easily emerge for various reasons (e.g. self-defeating arguments, reciprocal defeaters, non-concordant sets of supporting arguments). Since managing infinite argumentation lines is regarded as undesirable, suitable restrictions are introduced in the formal definition of argumentation line to avoid these cases.

A different kind of restriction is adopted in the argumentation-based approach to *Defeasible Logic* proposed in [54]. Here, an argument is a possibly infinite proof tree for a literal p (associated with the tree root). By definition, however, only finite arguments can be acceptable (this rather drastic choice is motivated by the goal of avoiding the risk of supporting "well-known fallacies such as circular argument and infinite regress") while infinite arguments keep anyway the power to prevent justification of other arguments.

In the context of the logic-based approach to argumentation of Besnard and Hunter [16] the classical chicken and egg dilemma is used as a common sense reasoning example giving rise to an infinite sequence of arguments, each being a counterargument to the preceding one (such a sequence is called *dispute* in this context). In fact, dilemmas (of various nature and possibly more interesting than chicken and egg) are recognized as a significant case of infinite reasoning with conflicting arguments also in non-technical literature.

An example is given in the novel "Runaround" by Isaac Asimov where, on the planet Mercury, a robot, called SPD-13, receives by two spacemen the order to accomplish a mission which requires to collect selenium from a pool. The robot is programmed to obey three basic rules which can be synthesized as follows. The first rule states that the robot has to protect human lives. The second rule states that the robot has to obey orders unless they conflict with the first rule. The third rule states that the robot has to protect itself unless this conflicts with the first rule. After a long wait, the spacemen send out another (less capable) robot to look for SPD-13. From the report of the second robot they realize that the mind of SPD-13 is in a loop (which has caused a sort of "drunkenness") which can be described as follows: when the robot gets near the selenium pool it perceives an unforeseen danger, the third rule is activated and the robot builds an argument to go away, prevailing on the previous decision to obey the order. When the robot is sufficiently far from the pool, its danger perception decreases and so according to the second rule it builds a new argument which leads it to turn back towards the pool, prevailing over the previous decision. When it gets sufficiently closer to the pool, it feels the danger again and the process restarts.

From the representation point of view in [16] this kind of problems is tackled by imposing some restriction in the definition of the *argument tree* structure, which is meant to capture all the disputes concerning a specific argument (represented by the root of the tree). More precisely, the premises of an argument added to the tree are forbidden to be a subset of the union of the premises of its ancestors. With this constraint, the argument tree for the chicken and egg dilemma reduces to a two-length chain, but, as observed in [16, p. 62] "the argument tree is merely a representation of the argumentation" and "altough the argument tree is finite, the argumentation here is infinite and unresolved".

An explicit restriction to finite structures is also adopted in a recent work concerning the study of postulates and properties of logic-based instantiations of abstract argumentation [53]. In this work, a propositional logic with a countable set of propositional letters is used as a basis for the argumentation process. It ensues that the set of all arguments is countably infinite (though, by definition, an argument is assumed to be built on a finite set of formulae). However, when introducing the notion of *argument graph*, where nodes are arguments and the arcs represent the attack relation, the authors restrict the consideration to graphs with a finite number of nodes.

2.2. Multi-agent systems

From the previous subsection it appears that while the potential existence of infinite structures is widely acknowledged in non-abstract argumentation contexts too, there is a prevailing attitude to overlook the difficult problem of actually managing them, by ascribing their genesis to undesirable/pathological conditions which can be avoided at the implementation level with proper programming and preventive checks on the knowledge base. While it is certainly true that infinite argumentation structures may arise from uninteresting/undesirable conditions, we remark that these do not exhaust the range of cases where such structures may arise and that, whatever the underlying reason, their emergence can not always be prevented. In fact, there are concrete situations where systematic well-founded argument generation mechanisms may incur in an open-ended non-terminating behavior.

Multi-agent (and more generally distributed) systems provide a major case for this statement, under the non restrictive and fairly standard assumptions of self-interest and absence of a global reasoner to which all information is available.

For instance in [17] argumentation semantics of Defeasible Logic is extended to the case of a multi-context system for distributed ambient intelligence. Each context corresponds to the local knowledge and reasoning of an agent and arguments of different contexts are interrelated through mapping rules. As to undesirable circularities, it is observed that "loops in the local knowledge bases can be easily detected and removed without needing to interact with other agents. However, even if there are no loops in the local theories, the global knowledge base may contain loops caused by mapping rules." Such loops in the global knowledge base may cause infinite argumentation lines. In [17] this problem is dealt with by adopting the specific assumptions that (i) each agent uses its own vocabulary and is therefore the unique responsible of the evaluation of some

literals, (ii) the agents behave in a fully cooperative manner in the process of justification evaluation.

In fact, unless one adopts some restrictive assumptions of this kind, the possible onset of non terminating behaviors in argument defeat status computation is inherent to multi-agent system. This is formally proved in [10] where it is shown that approaches to distributed defeat status computation (see for instance [61, 70]) usually rely on assumptions like a predefined unmodifiable number of agents, the existence of a centralized structure, or the obligation to reveal the entire inner structure of the arguments an agent has built. Removing these restrictions and assuming a multi-agent system with the general properties of *unlimited cardinality, autonomy, asynchronism, dynamism, and uncostrained communication* (see [10] for details), an impossibility result is obtained showing that no self-stabilizing algorithm can exist for defeat status computation according to any semantics which is *valid* (namely obeys some fairly general constraint on the defeat status assignment). The paper provides two practical examples of non terminating behavior: a distributed version of the three liars paradox introduced by Pollock [73] and a negotiation dialogue for resource exchange among three agents[2].

In the following subsections we provide two extended examples of infinite argumentation in multi-agent systems, namely an infinite negotiation process and a distributed reasoning process involving the components of an ambient intelligence system. These examples are inspired to the application contexts considered in [10] and [17] and will be used in Section 7 to demonstrate the application of the formalism proposed in this paper.

2.2.1. An example in multi-agent negotiation

In multi-agent systems, independent and possibly self-interested components strive to achieve common or individual goals by various forms of interaction (cooperation, negotiation, persuasion, resource exchange, task allocation, ...) for most of which argumentation is considered a suitable model in the literature (see for instance [78] and the references thereof).

If one considers interactions involving more than two agents and removes some not always realistic assumptions (e.g. that information on all argument exchanges is available to all agents) the interaction process may not reach a solution and continue forever with an infinite production of arguments.

To exemplify, consider a simple negotiation setting where three agents A_1, A_2 and A_3 may exchange resources called R_a, R_b, R_c. Each agent possesses some resource and has its own preference ordering on resources. Each agent is only partially informed on the resources owned by other agents and can not know the preferences of other agents. At a given time instant, an agent A_x builds an

[2]By the way, in the context of this example, in [10] it is remarked that the existence of circularities at the global level is not critical *per se*, since they do not give rise to any problem if there are further attacking arguments breaking the cycle. Thus, simply forbidding cycles turns out to be a too drastic measure in general.

argument for proposing to A_y an exchange between resources R_x and R_y if the following conditions holds: (i) A_x owns R_x; (ii) A_x knows that A_y owns R_y; (iii) A_x prefers R_y to R_x. An agent receiving a proposal may accept or reject it at a later time: the agent who has sent the proposal is free to withdraw it before receiving confirmation of acceptance (typically because the agent has received a more convenient exchange proposal which conflicts with the previous one). An agent is free to reiterate a proposal after having withdrawn it (typically because the reason to withdraw does not hold any more) and is obliged to withdraw an offer s/he has made before accepting an incompatible offer s/he has received. Message exchanges between two agents are not available to other agents but they are collected by an authority supervising the negotiation arena. The authority is informed on all the exchanged messages and on the resources possessed by all agents but has not access to agents' preference rankings. The authority may therefore build an argumentation framework representing the evolution of the negotiation process and may help agents to overcome critical situations. We suppose that the attack relation in the argumentation framework managed by the authority is defined on the basis of the two following rules:

- a (possibly reiterated) proposal P_1 (received or sent by an agent A_i) attacks a (possibly reiterated) proposal P_2 (received or sent by the same agent A_i) if accepting the exchange proposed in P_1 makes impossible the exchange proposed in P_2;
- a withdrawal obviously attacks the withdrawn proposal, a reiterated proposal attacks the corresponding previous withdrawal.

Suppose now that the initial situation is the one described in Table 1.

Agent ID	Owns	Knows	Preference rank
A_1	R_c	A_2 owns R_b	$R_a > R_b > R_c$
A_2	R_b	A_3 owns R_a	$R_c > R_a > R_b$
A_3	R_a	A_1 owns R_c	$R_b > R_c > R_a$

Table 1: Initial situation of the negotiation example.

Then each agent builds an offer as follows:

- A_1 sends an offer to A_2 proposing an exchange between R_c and R_b: $O_1 = Off(t_0, (A_1, A_2, Exch(R_c, R_b)))$
- A_2 sends an offer to A_3 proposing an exchange between R_b and R_a: $O_2 = Off(t_0, (A_2, A_3, Exch(R_b, R_a)))$
- A_3 sends an offer to A_1 proposing an exchange between R_a and R_c: $O_3 = Off(t_0, (A_3, A_1, Exch(R_a, R_c)))$

Clearly each offer is incompatible with the two others.

It can be seen that each agent prefers the status resulting from the exchange in the offer s/he has received wrt the one resulting from the offer s/he has

made. For instance, agent A_1 prefers exchanging R_c with R_a (as proposed by A_3) than exchanging R_c with R_b (as s/he has proposed to A_2). Then, let say at time t_1, each agent sends a message of withdrawal of the previous offer: $W_1 = Wd(t_1, (A_1, A_2, Exch(R_c, R_b)))$, $W_2 = Wd(t_1, (A_2, A_3, Exch(R_b, R_a)))$, $W_3 = Wd(t_1, (A_3, A_1, Exch(R_a, R_c)))$.

As a consequence of the withdrawal and of its local view, each agent is now, let say at time t_2, in a position where the only reasonable move is to reiterate the initial offer (let say that these messages are denoted as O_4, O_5, O_6): clearly this reproduces the initial situation, causes three further withdrawals and the process goes on forever[3].

2.2.2. An example in ambient intelligence

Consider a system of ambient intelligence consisting of several independently developed interacting components, some of which join and leave dynamically the system, as described in [17].

Adapting an example presented in [17] suppose that the system includes the following components:

- a people locator;
- a video surveillance system for each room;
- a lighting management system for each room;
- personal smartphones.

The components interact as follows:

- personal smartphones notify their position to the people locator;
- the video surveillance system notifies the results about people detection to the people locator;
- the lighting management system has a light sensor and informs the video surveillance system whether each room is dark or not;
- the people locator informs the lighting system about people's presence in each room.

The people locator uses the following rules:

R1: if a smartphone is in a room the smartphone owner is in the room

[3]Note that a similar situation occurs also if we assume that each agent updates its knowledge on who owns what after the first round of offers. In that case the roles of bidder and addressee would be interchanged (e.g. the exchange of R_b with R_c would be proposed by A_2 to A_1 and so on), but the non-terminating sequence of offers and withdrawals would occur in the same way.

R2: if the video surveillance notifies the presence of a person in a room then there is a person in the room

R3: if the video surveillance notifies the absence of any person in a room then there is no person in the room

R4: if a person is present in a room at time t then the person is present in the room at time (t+1)

R5: if a person is not present in a room at time t then the person is not present in the room at time (t+1)

The video surveillance system uses the following rules:

R6: if it is not dark and the image processing system recognizes a person in a room then the video surveillance system notifies the presence of a person in the room

R7: if it is not dark and the image processing system does not recognize any person in a room then the video surveillance system notifies the absence of any person in the room

The lighting system uses the following rules:

R8: if it is dark in a room and a person is in the room switch the room lights on

R9: if the lights are on in a room and no person is in the room switch the room lights off

The people locator uses two default persistence rules (R4 and R5) which can be applied in absence of new information and are the weakest ones: in case of conflicting conclusions those derived using R4 and R5 are overruled by those derived using R1, R2, and R3. Moreover video surveillance is regarded as providing more reliable information than the mere presence of a smartphone, hence R3 is stronger than R1.

To make the presentation compact, let us omit the details concerning message exchanges among the various components and, as a consequence, combine rules together where possible. The set of rules presented above can be represented by the following logic program[4] M, where *not* denotes *negation as failure* and $\neg$ denotes *explicit* negation.

[4]The program, with the restriction to a finite time horizon, has been run in DLV.

M :

$in(p,r,t)$	$\leftarrow phone(x), person(p), owner(x,p), room(r), phonein(x,r,t),$ $not\ videovalid(r,t)$	$(r1)$
$in(p,r,t)$	$\leftarrow room(r), person(p), videorecogn(p,r,t), videovalid(r,t)$	$(r2)$
$\neg in(p,r,t)$	$\leftarrow room(r), person(p), \neg videorecogn(p,r,t), videovalid(r,t)$	$(r3)$
$in(p,r,s(t))$	$\leftarrow person(p), phone(x), room(r), owner(x,p), in(p,r,t),$ $not\ videovalid(s(t)), not\ phlocated(x,s(t))$	$(r4)$
$\neg in(p,r,s(t))$	$\leftarrow person(p), phone(x), room(r), owner(x,p), \neg in(p,r,t),$ $not\ videovalid(r,s(t)), not\ phlocated(x,s(t))$	$(r5)$
$lighton(r,s(t))$	$\leftarrow room(r), person(p), in(p,r,t), dark(r,t)$	$(r6)$
$\neg lighton(r,s(t))$	$\leftarrow room(r), person(p), \neg in(p,r,t)$	$(r7)$
$phlocated(x,t)$	$\leftarrow phone(x), room(r), phonein(x,r,t)$	$(r8)$
$videovalid(r,t)$	$\leftarrow not\ dark(r,t), room(r)$	$(r9)$

Most predicates in M have self-explaining names. We assume that variable t refers to time instants, which are discrete and totally ordered, and that $s(t)$ denotes the successor of instant t. We assume that information about the presence of smartphones, darkness, and the outcome of the video recognition system is available for each room in the form of asserted or explicitly negated facts at each time instant, as provided by the relevant devices. In particular, we assume that the image processing component returns either *videorecogn* or $\neg$*videorecogn* for each triple (p,r,t) and, of course, does not recognize any person when it is dark. We also assume that all devices are properly working so that, in particular, when light is on in a room r at time t the predicate $dark(r,t)$ is false.

R1 is represented by line $(r1)$ of M with the condition $not\ videovalid(r,t)$ to ensure priority to the video surveillance system when its output is valid, namely when it is not dark in the room, as specified by $(r9)$. Line $(r2)$ synthesises rules R2 and R6, and, similarly, $(r3)$ synthesises R3 and R7. The persistence rules R4 and R5 are represented by lines $(r4)$ and $(r5)$ with the conditions $not\ videovalid(s(t))$ and $not\ phlocated(s(t))$ to ensure that other rules prevail when information from devices is available and valid (for phones $(r8)$ applies, where the predicate $phlocated(x,t)$ means that the location of phone x, whatever it is, is known at instant t). Rules R8 and R9, concerning the lighting system are represented respectively by lines $(r6)$ and $(r7)$.

Suppose now that Brian at time instant 0 (when outside is dark) exits the office, switches the light off and forgets his smartphone.

It follows that, applying $(r1)$, $in(Brian, office, 0)$ is derived and as a consequence, by $(r6)$, light is switched on at instant 1. Then the room is no more dark, $videovalid(office, 1)$ holds by $(r9)$ and since $\neg videorecogn(Brian, office, 1)$ holds, by $(r3)$ $\neg in(Brian, office, 1)$ is derived and, applying $(r7)$, the light is switched off. As a consequence at instant 2 the room is dark and $videovalid(office, 2)$ can not be derived. $(r1)$ then applies and $in(Brian, office, 2)$ is derived, it follows that, by $(r6)$, light is switched on at instant 3, and so on.

2.2.3. Non-cooperative dialogues

To avoid situations of the kind described in Sections 2.2.1 and 2.2.2, most argumentation-related dialogue protocols in the literature (see for instance [90,

88, 42, 75, 62]) concern the two-party case, which implies in particular that all moves are known to all dialogue participants, and assume that both participants accept some rules (in particular some kind of non repetition constraint) in order to guarantee termination[5]. While, as already remarked, this guarantee can not be extended to more general contexts, it can also be observed that works encompassing non-cooperative, and hence potentially infinite, two-party dialogues have been considered in the literature (see for instance [51, 50]). In particular, in [51] the authors describe several kinds of non-cooperative dialogue games, such as so-called *stone-walling* tactics. An example [51][p.178] is the following game between two agents, the proponent (P) and the opponent (O):

Example 1. Example of stone-walling:

m_1: **Assert** α (P)
m_2: **Reject** α (O)
m_3: **Assert** α (P)
m_4: **Assert** β (O)
m_5: **Argue since** α, **either** $\neg\beta$ **or** $\beta \nvdash \neg\alpha$ (P)
m_6: **Assert** λ (O)
m_7: **Argue since** α, **either** $\neg\lambda$ **or** $\lambda \nvdash \neg\alpha$ (P)
m_8: ...

P makes substantially the same move from m_5 onwards: this can be interpreted as being convinced that α cannot be false and that no case against α can succeed. This kind of non-cooperation structure is called *mind closed* and it is easy to imagine such a dialogue continuing forever if P's mind does not change.

In the previous example, stone-walling is done by one party, P. In [51][p.181], however, the authors define a quarrel as a reciprocal stone-walling dialogue. As the authors remark, the quarrel model shows up frequently in actual dialogical practice, and they suggest that this is an efficient way of playing the game of *dialectical fatigue*. Dialectical fatigue settles a dispute, and declares a win for the party whose opponent just gives up[6]:

Example 2. Example from [51][p.182]:

m_1: **Assert** α (P)
m_2: **Reject** α (O)
m_3: **Assert** α (P)
m_4: **Reject** α (O)
m_5: ...
m_n: **Assert** α (P)

[5] In [88] it is remarked however that the case of infinite proofs is problematic and is left for future developments.

[6] Though not explicitly mentioned, the notion of "dialectical fatigue" and its exploitation by self-interested agents underpins the examples discussed in [39].

m_{n+1}: **Perhaps you are right** (O)

One may wonder what the outcome of the dialogues in Examples 1 and 2 ought to be from a computational perspective. If the traditional *termination rule* is adopted, the outcome is that the proponent wins if the opponent concedes the main claim, and the opponent wins if the proponent retracts the main claim [75]. However, this rule can not be applied to the non-terminating Example 1. Moreover termination in Example 2 is due to fatigue of one of the players, so that this outcome may be regarded as "non-rational", while a "rational" development of Example 2 would be non-terminating too. One may also observe that the termination rule imposes that one position prevails over the other one, while one might consider also the case where the two positions are regarded as equally acceptable.

Besides issues concerning dialogues, multi-agent systems provide a further case for non terminating argumentation in presence of reasoning about mutual beliefs. In fact an agent $ag1$ able to reason about the beliefs of another agent $ag2$, may take into account also the ability of $ag2$ to reason about the beliefs of $ag1$ in turn, then both $ag1$ and $ag2$ may reason about the mutual beliefs about beliefs an so on *ad libitum*. This kind of problem is exemplified, in a common sense setting, by the novel by the Argentine writer Osvaldo Soriano "The longest penalty ever" [83]. Here, in a football game, a goal keeper has to reason about whether to dive to the left or to the right. The keeper knows that the kicker in the past has always kicked to the right: this would be a reason to dive to the right, but the keeper also knows the kicker knows that the keeper knows his past records (and might therefore decide to kick to the left) and this would be a reason to dive to the left, but in turn the kicker knows that the keeper knows that the kicker knows ..., and the chain of mutually attacking arguments supporting the decision of diving to the left or to the right grows to infinity. Of course, in a non cooperative context this growth can not be prevented by mutual agreement and, for either agent, stopping the reasoning at a given level represents an arbitrary, and possibly not appropriate, choice.

2.3. Reasoning with unbounded domains

Leaving apart non-terminating situations in multi-agent systems, a further example of infinite argumentation concerns reasoning with unbounded domains like time or space.

For instance, reactive systems are characterized by "their perpetual interaction with their environment as well as their nonterminating behaviour" [55] and as such require models able to encompass infinite objects like automata over infinite words or infinite games. While these models are suitable to analyze properties of these systems in a *monotonic* reasoning context, different issues arise and different formalisms are needed in case some kind of nonmonotonic reasoning is carried out.

An example of use of argumentation in a nonmonotonic open-horizon context is provided by Pollock [74], who introduced a *temporal projection principle* to address the problem of argumentation-based reasoning on "stable" properties

of the world. Quoting Pollock "the built-in epistemic arsenal of a rational agent must include reason-schemas of the following sort for at least some choices of P: if $t_0 < t_1$, believing $P - at - t_0$ is a defeasible reason for the agent to believe $P - at - t_1$". To allow new information to override presumptions based on out-of-date perceptions it is necessary that "the strength of the presumption that a stable property will continue to hold over time decays as the time interval increases". Though not explicitly addressed by [74], it is straightforward to consider a spatial version of this projection principle too. For instance if one has a reason to believe that a certain site is highly dangerous due to pollution or contamination, then this belief can be reasonably projected to all the neighbour locations with strength decaying as the distance from the contaminated site increases. The set of arguments (with different strength) that can be produced on the basis of this kind of projection principles is, in general, unbounded. To cope with this, in the OSCAR implementation described in [74], Pollock restricts the use of the temporal projection principle to a specific form of backward reasoning: the agent is interested in the value of a property at a specific time instant t and checks whether there are reasons to believe that the property had a certain value at an instant $t_0 < t$. If this is the case, the reason can be projected, with decreased strength, from t_0 to t. However explicit representations of infinite arguments are needed to go beyond this specific form of reasoning.

Formulating (defeasible) previsions on the basis of (discrete) series of past observations is a further form of reasoning involving similar issues as the set of possible observations is countably infinite and the set of actual observations may, in general, grow indefinitely. To give an example, consider previsions concerning sport events (e.g. soccer matches) based on previous performances of the teams with defeasible rules of the kind "A team which has won the majority of past matches will win future matches" and "A team which has lost the last three matches will loose next match" (or more complex ones with similar structure). Here the observation of the outcomes for a team is constantly updated after each match giving rise to new arguments representing new previsions (possibly conflicting each other and with previous ones). The set of generated "previsional" arguments is (at least in principle) infinite at each step as some of the previsions could be projected over the set of all future matches, e.g. for a very strong team with very long tradition one can, reasonably but defeasibly, foresee further wins for many years to come. It can be observed that open-ended horizons of this kind can be managed in practice by considering a finite temporal window excluding time instants which are "too far" in the past or in the future. It can be also observed, however, that if such a temporal window is very large it can be anyway more convenient in practice to adopt compact specification techniques for infinite frameworks of the kind we propose in this paper rather than to deal explicitly with all the elements of finite (but very large) sets and that, in any case, an open-ended representation is more appropriate for reasoning concerning long-term trends and scenarios.

As already mentioned, unbounded time horizons have been considered also in game theory, where infinite games are meant to represent open-ended (e.g. life-long) interactions between the players. Different kinds of infinite games can

be considered. In iterated variable-sum non-cooperative games, like the iterated prisoner dilemma [3], each player has a payoff at each turn and seeks strategies maximizing the value of the infinite series of the payoffs. Quite differently, in Gale-Stewart two-player zero-sum games, one of the two player wins depending on the membership of the infinite sequence of the moves played by both players to a predefined *payoff* set. Here a winning strategy for a player is a function for choosing the next move which ensures the membership (or non-membership) of the resulting infinite sequence to the payoff set. There is a potential rich interplay between infinite games and infinite argumentation. On one hand, infinite argumentation frameworks can be used as an abstract model for some game-theoretic problems and, especially thanks to their rich endowment of alternative semantics, may suggest variants and open new perspectives for these problems, in the same spirit as done by Dung for the stable marriage problem (see [37, Sect. 3.2]). On the other hand existing results on infinite games may provide a formal basis for the open investigation area on infinite argumentation games and the relevant strategies, building on the standpoint that non-termination does not mean necessarily indeterminacy.

2.4. Motivation summary

Summing up, it appears that investigation on infinite argumentation structures got somehow stuck in a sort of deadlock situation. From a theoretical point of view, their fundamental role has been consistently acknowledged and they have been universally encompassed at a definitional level, but actual formalisms to deal with them at an operational level, i.e. compact representations along with computational procedures, have not been developed, possibly due to a "lack of pressure" from the application side. In turn, the potential emergence of infinite argumentation structures has been evidenced in a variety of application contexts, but, possibly due to the lack of suitable operational approaches from the theoretical side, they have generally been disregarded as problematic or dealt with by adopting specific workarounds.

The present work contributes to overcome this situation by proposing an approach to compact specification of infinite abstract argumentation frameworks endowed with effective computational procedures.

The approach is suitable, in general, to describe infinite argumentation frameworks with some kind of regular structure. This covers, in fact, the cases of practical interest, since it corresponds to the generation of arguments (and of the attacks between them) by some systematic non terminating mechanism, as it may occur in a multi-agent system or in other automated reasoning contexts, as described above. As it will be better commented later, the approach is also suitable to manage cases where argument generation terminates but the resulting framework is so large to make a compact representation advantageous. Our proposal can therefore be regarded as a novel enabling technique with respect to the long-term goal of deploying extended argumentation-based reasoners covering also the case of infinite (or very large) frameworks. In a shorter-term perspective, the results in this paper provide a formal basis for incorporating

the management of infinite frameworks in existing implementations of Dung's style argumentation like ASPARTIX [47] or Dungine [84].

3. Background notions

3.1. Argumentation frameworks

In this section we review the elements of Dung's abstract argumentation frameworks [37] and the relevant semantics notions and basic computational issues.

Definition 1. An *argumentation framework* (AF) is defined as a pair $\langle \mathcal{X}, \mathcal{A} \rangle$ in which $\mathcal{X}$ is a set of *arguments* and $\mathcal{A} \subseteq \mathcal{X} \times \mathcal{X}$ describes the *attack relation* between arguments in $\mathcal{X}$, so that $\langle x, y \rangle \in \mathcal{A}$ indicates "the argument x *attacks* the argument y" (or, equivalently, "the argument y *is attacked by* the argument x").

For $S \subseteq \mathcal{X}$ we use the notations S^- (resp. S^+) to indicate

$$\begin{array}{rcl} S^- & = & \{\ x \in \mathcal{X}\ :\ \exists\ y \in S \text{ for which } \langle x, y \rangle \in \mathcal{A}\} \\ S^+ & = & \{\ x \in \mathcal{X}\ :\ \exists\ y \in S \text{ for which } \langle y, x \rangle \in \mathcal{A}\} \end{array}$$

The arguments in S^- (resp. S^+) are said to attack (resp. be attacked by) S.

When for any argument $x \in \mathcal{X}$, the set $\{x\}^-$ of its attackers is finite, the argumentation framework is said to be *finitary*. Formally, an AF, $\langle \mathcal{X}, \mathcal{A} \rangle$, is *finitary* iff for each argument $x \in \mathcal{X}$ $|\{\ y\ :\ \langle y, x \rangle \in \mathcal{A}\}|$ is finite.

A subset $S \subseteq \mathcal{X}$ is *conflict-free* if no argument in S attacks another argument in S, i.e. $(S \times S) \cap \mathcal{A}$ is empty. An argument x is said to be *acceptable* with respect to $S \subseteq \mathcal{X}$ if for any $y \in \mathcal{X}$ such that $\langle y, x \rangle \in \mathcal{A}$ there is some $z \in S$ for which $\langle z, y \rangle \in \mathcal{A}$, i.e. x is acceptable wrt to S if any attacker (y) of x is counterattacked by an argument (z) of S.

The *characteristic function* of an AF is the mapping $\mathcal{F}\ :\ 2^{\mathcal{X}}\ \rightarrow\ 2^{\mathcal{X}}$ where

$$\mathcal{F}(S)\ \ =\ \ \{\ x \in \mathcal{X}\ :\ x \text{ is acceptable wrt } S\}$$

Much of the development of AFs has focused on the study of *argumentation semantics* which can be regarded as refining the informal idea of "collection of justifiable arguments in an AF". Typically this has been achieved by considering predicates that such collections must satisfy, i.e. mappings $\sigma\ :\ 2^{\mathcal{X}} \rightarrow \{\top, \bot\}$ so that $\mathcal{E}_\sigma(\langle \mathcal{X}, \mathcal{A} \rangle)$ describes the set of subsets of $\mathcal{X}$ that satisfy the criteria given by σ within the AF $\langle \mathcal{X}, \mathcal{A} \rangle$. A review of the many choices that have been considered for σ may be found in Baroni and Giacomin [7].

Definition 2. Let $\langle \mathcal{X}, \mathcal{A} \rangle$ be an AF and S a subset of $\mathcal{X}$.

a. S is *admissible* (denoted as $S \in \mathcal{E}_{adm}(\langle \mathcal{X}, \mathcal{A} \rangle)$) if S is conflict-free and every argument in S is acceptable wrt S, i.e. $S \subseteq \mathcal{F}(S)$.
b. S is a *complete extension*, (denoted as $S \in \mathcal{E}_{comp}(\langle \mathcal{X}, \mathcal{A} \rangle)$) if S is conflict-free and $x \in S$ if and only if x is acceptable wrt S, i.e. $S = \mathcal{F}(S)$.

c. S is a *preferred extension* (denoted as $S \in \mathcal{E}_{pr}(\langle \mathcal{X}, \mathcal{A} \rangle)$) if S is a *maximal* (wrt $\subseteq$) admissible set.
d. S is a *stable extension* (denoted as $S \in \mathcal{E}_{stab}(\langle \mathcal{X}, \mathcal{A} \rangle)$) if S is conflict-free and for any $y \notin S$, there is some $x \in S$ that attacks y, i.e. $S^+ = \mathcal{X} \setminus S$.
e. S is the *grounded extension* of $\langle \mathcal{X}, \mathcal{A} \rangle$ (denoted as $S \in \mathcal{E}_{gr}(\langle \mathcal{X}, \mathcal{A} \rangle)$) if it is the (unique) least fixed point of $\mathcal{F}$, i.e. $S = \mathcal{F}(S)$ and there is no $S' \subsetneq S$ such that $S' = \mathcal{F}(S')$.

The existence and uniqueness of the grounded extension is established in [37] for all AFs.

Finally, we recall the various ways in which a given argument may relate to these sets in an AF $\langle \mathcal{X}, \mathcal{A} \rangle$.

Definition 3. Let $x \in \mathcal{X}$ and $\sigma : 2^{\mathcal{X}} \to \{\top, \bot\}$. The argument x is *credulously accepted* wrt σ if there is some S in $\mathcal{E}_\sigma(\langle \mathcal{X}, \mathcal{A} \rangle)$ such that $x \in S$. It is said to be *sceptically accepted* wrt σ if every S in $\mathcal{E}_\sigma(\langle \mathcal{X}, \mathcal{A} \rangle)$ satisfies $x \in S$.

The concepts of credulous and sceptical acceptance, together with the various semantics that have been put forward, naturally motivate a number of computational problems involving AFs.

Definition 4. Let $\sigma : 2^{\mathcal{X}} \to \{\top, \bot\}$.

a. CA_σ is the decision problem whose instances, $\langle \langle \mathcal{X}, \mathcal{A} \rangle, x \rangle$, are accepted if and only if x is credulously accepted wrt σ in $\langle \mathcal{X}, \mathcal{A} \rangle$.
b. SA_σ is the decision problem whose instances, $\langle \langle \mathcal{X}, \mathcal{A} \rangle, x \rangle$, are accepted if and only if x is sceptically accepted wrt σ in $\langle \mathcal{X}, \mathcal{A} \rangle$.
c. VER_σ is the decision problem whose instances, $\langle \langle \mathcal{X}, \mathcal{A} \rangle, S \rangle$, are accepted if and only if $S \in \mathcal{E}_\sigma(\langle \mathcal{X}, \mathcal{A} \rangle)$.
d. EXIST_σ is the decision problem whose instances, $\langle \mathcal{X}, \mathcal{A} \rangle$, are accepted if and only if $\mathcal{E}_\sigma(\langle \mathcal{X}, \mathcal{A} \rangle) \neq \emptyset$.
e. NON-EMPTY_σ is the decision problem whose instances, $\langle \mathcal{X}, \mathcal{A} \rangle$, are accepted if and only if $\mathcal{E}_\sigma(\langle \mathcal{X}, \mathcal{A} \rangle) \notin \{\emptyset, \{\emptyset\}\}$.

As well as the *decision problems* described in Defn. 4 there are a range of *function* (or *construction*) problems. We focus on that of given $\langle \mathcal{X}, \mathcal{A} \rangle$ (for which $\mathcal{E}_\sigma(\langle \mathcal{X}, \mathcal{A} \rangle) \neq \emptyset$) identifying all sets $S \subseteq \mathcal{X}$ for which $S \in \mathcal{E}_\sigma(\langle \mathcal{X}, \mathcal{A} \rangle)$, denoting this problem CONS_σ.

For each of the semantics σ presented in Defn. 2, these computational problems have been studied in depth (within *finite* AFs) and their general properties are now well understood. We summarise these results in Fact 1 and Table 2.

Fact 1.

a. The function CONS_{gr} is polynomial time computable, hence all of the cases (a) though (e) of Defn. 4 are polynomial time decidable for the grounded extension [37].
b. For $\sigma \in \{adm, pr, comp, gr\}$, EXIST_σ is trivial (i.e. it is always verified as a consequence of well-known results).

c. EXIST$_{stab}$ is NP–complete [36] (see also [49]).

d. For $\sigma \in \{adm,\ stab,\ comp\}$, VER$_\sigma$ is decidable in polynomial-time, however VER$_{pr}$ is coNP–complete [36].

e. For $\sigma \in \{adm,\ pr,\ stab,\ comp\}$, CA$_\sigma$ is NP–complete [36].

f. SA$_\sigma$ is trivial for $\sigma = adm$, polynomial for $\sigma = comp$, coNP–complete for $\sigma = stab$[7], and Π_2^p–complete for $\sigma = pr$ [41].

g. NON-EMPTY$_\sigma$ is NP–complete for $\sigma \in \{adm,\ pr,\ comp,\ stab\}$ [36].

	$\sigma = adm$	$\sigma = pr$	$\sigma = comp$	$\sigma = stab$	$\sigma = gr$
EXIST$_\sigma$	trivial	trivial	trivial	NP–complete	trivial
VER$_\sigma$	polynomial	coNP–complete	polynomial	polynomial	polynomial
CA$_\sigma$	NP–complete	NP–complete	NP–complete	NP–complete	polynomial
SA$_\sigma$	trivial	Π_2^p	polynomial	coNP–complete	polynomial
NON-EMPTY$_\sigma$	NP–complete	NP–complete	NP–complete	NP–complete	polynomial

Table 2: Computational problems in finite AFs.

We emphasise that the classifications in Fact 1 are with respect to finite AFs. For a more detailed summary of complexity and algorithms within AF semantics we refer the reader to the overview of Dunne and Wooldridge [44]; complexity-theoretic treatments of both novel semantics and developments of Dung's original proposals may be found in, among others, [6, 46, 40, 43, 45].

3.2. Formal languages

As to the required background on formal languages, which will be heavily used in the paper, we assume that the reader is familiar with the standard concepts and basic results in the field (to make the paper self-contained the necessary ones are provided in Appendix A). We recall only the basic definitions in this section, in order to introduce the reader to the notation used in the sequel of the paper.

Definition 5. An *alphabet* is a *finite* set of symbols. For an arbitrary alphabet, the notation $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ will be used. A *word*, w, over an alphabet Σ is a *finite sequence*, $w = w_{i_1} w_{i_2} \cdots w_{i_r}$ of symbols from Σ. The set of all possible words is denoted as Σ^*. The *length*, $|w|$, of $w \in \Sigma^*$ is the total number of symbols occuring in its definition. The word of length 0 in Σ^* is called the *empty word* and is denoted as ε.

For $u = u_{i_1} \cdots u_{i_r}$ and $v = v_{j_1} \cdots v_{j_s}$ words in Σ^* the word $w \in \Sigma^*$ formed by *concatenating* u with v (denoted $u \cdot v$) is the word $u_{i_1} \ldots u_{i_r}\ v_{j_1} \ldots v_{j_s}$ whose length is $|u| + |v| = r + s$. For any $u \in \Sigma^*$, $u \cdot \varepsilon = \varepsilon \cdot u = u$, i.e. ε is an identity element in Σ^* with respect to the operation $\cdot$ of concatenation.

[7] Note, however, the *caveat* raised in [44].

Definition 6. A *language*, L, over an alphabet Σ, is a subset of Σ^*. For languages L_1 and L_2 we define languages $L_1 \cup L_2$, $L_1 \cap L_2$, and $L_1 \setminus L_2$ in the obvious way so that the operations $\{\cup, \cap, \setminus\}$ are the standard set-theoretic ones.

In addition, specific to languages,

$$\begin{array}{lcl} L_1 \cdot L_2 & = & \{u \cdot v \ : \ u \in L_1, \ v \in L_2\} \\ \overline{L} & = & \{u \ : \ u \in \Sigma^*, \ u \notin L\} \\ L^* & = & \cup_{k=0}^{\infty} \{ \ w \ : \ w = u_1 \cdot u_2 \cdots u_k, \ u_i \in L\} \\ L_1/L_2 & = & \{ \ u \ : \ \exists \ v \in L_2 \text{ s.t. } u \cdot v \in L_1\} \\ rev(L) & = & \{ \ \sigma_1\sigma_2 \cdots \sigma_{m-1}\sigma_m \ : \ \sigma_m\sigma_{m-1} \cdots \sigma_2\sigma_1 \in L\} \end{array}$$

The language L^* is sometimes referred to as the *Kleene closure* (or $*$-closure) of L, while L_1/L_2 is called the *quotient* of L_1 wrt L_2.[8]

Definition 7. A language $L \subseteq \Sigma^*$ is a *regular language* if L satisfies any of the following requirements:

R1. $L = \emptyset$ or $L = \{\varepsilon\}$ or $L = \{\sigma\}$ for any $\sigma \in \Sigma$.

R2. $L \ = \ L_1 \cup L_2$ where L_1 and L_2 are regular languages.

R3. $L \ = \ L_1 \cdot L_2$ where L_1 and L_2 are regular languages.

R4. $L = (L_1)^*$ where L_1 is a regular language.

Definition 8. A *formal grammar* is defined via a 4-tuple, $\langle \Sigma, V, P, S \rangle$ where $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ is a *finite alphabet* of *terminal symbols*; $V = \{V_1, \ldots, V_m\}$ a finite set of *variable symbols*, P is a finite set of *production rules*, $\{p_1, p_2, \ldots, p_r\}$ of the form $p_i \ : \ \alpha_i \to \beta_i$ where $\alpha_i \in (V \cup \Sigma)^* \setminus \Sigma^*$ and $\beta_i \in (V \cup \Sigma)^*$ and $S \in V$ is the *start symbol*. The language generated (see Appendix A)) by a grammar G is denoted as $L(G)$.

Definition 9. A *deterministic finite automaton* (DFA) is defined via a 5-tuple, $M = \langle \Sigma, Q, q_0, F, \delta \rangle$ where $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ is a finite set of input symbols, $Q = \{q_0, q_1, \ldots, q_m\}$ a finite set of *states*; $q_0 \in Q$ the *initial* state; $F \subseteq Q$ the set of *accepting* states; and $\delta \ : \ Q \times \Sigma \ \to \ Q$ the *state transition* function. A word $w = w_n w_{n-1} \ldots w_1 \in \Sigma^*$ is accepted by the DFA $\langle \Sigma, Q, q_0, F, \delta \rangle$ if the sequence of states $q_{i_1} q_{i_2} \ldots q_{i_n}$ consistent with the state transition function δ which processes every symbol in w, i.e. satisfying $q_{i_1} = \delta(q_0, w_1)$ and $q_{i_j} = \delta(q_{i_{j-1}}, w_j)$ for each $2 \leq j \leq n$, has $q_{i_n} \in F$. For a DFA, $M = \langle \Sigma, Q, q_0, F, \delta \rangle$, $L(M)$ is the subset of Σ^* accepted by M.

[8] Some authors distinguish so-called *left* and *right* quotients of L_1 wrt L_2, the latter being L_1/L_2 (as given in the definition), the former $\{ \ u \ : \ \exists v \in L_1 \text{ s.t. } v \cdot u \in L_2\}$. We use only the notion of (right) quotient.

4. Formalism requirements and weaknesses of naive representations

Given the goal of investigating novel approaches to deal with argumentation frameworks with a countably infinite set of arguments, we need to establish some basic criteria to evaluate the approaches themselves.

A first basic criterion is *expressiveness*, namely the ability to encompass the description of a sufficiently large variety of infinite argumentation frameworks so as to cover those cases which are meaningful from a theoretical or practical perspective. These include in particular the cases of infinite argumentation frameworks already considered in the literature.

A second criterion is *tractability*: the use of the formalism should not raise intractable computational problems making it impractical. In particular, we have to notice the arousal of a problem not occurring in the finite case: given that an argumentation framework $\langle \mathcal{X}, \mathcal{A} \rangle$ involving infinite sets can only be given through a finite encoding $\eta(\langle \mathcal{X}, \mathcal{A} \rangle)$, it must be validated that an encoding $\eta(\langle \mathcal{X}, \mathcal{A} \rangle)$ is indeed a valid description of *some* AF.

Further, computational requirements related to the basic decision problems listed in Definition 4 have to be taken into account. A third criterion is therefore *closure* wrt set-theoretical operations, as they are involved in the definition and/or characterization of the fundamental properties in argumentation semantics and hence in the relevant decision procedures. To exemplify, testing whether a set of arguments is conflict-free corresponds to test whether the intersection between this set and the set of its attackers is empty. Hence, given the specification of two infinite sets of arguments in a formalism, the specification of their intersection should be captured (and, hopefully, be easily constructable) within the same formalism.

In the view of satisfying the above requirements, a standard approach to the problem of representing an infinite collection of objects via a finite specification is to exploit formal grammars and their associated machine models[9].

In this context, we now consider and criticize a rather straightforward approach one might adopt in describing $\langle \mathcal{X}, \mathcal{A} \rangle$ where the supporting set of arguments is an infinite, but enumerable, set. The idea, introduced in Definition 10 consists in describing the attack relation with a language referring to indexes in the argument enumeration.

Definition 10. Let $\mathcal{X} = \{x_1, x_2, \ldots, x_n, \ldots\}$ be a countably infinite set of atomic arguments. A subset $\mathcal{A} \subseteq \mathcal{X} \times \mathcal{X}$ is *naively encoded* if described as the language $L_{\mathcal{A}}$ over the two symbol alphabet $\{0, 1\}$ for which

$$L_{\mathcal{A}} \quad = \quad \{0^i \cdot 1 \cdot 0^j \ : \ \langle x_i, x_j \rangle \in \mathcal{A}\}$$

Thus the naive encoding of a set of attacks uses a unary[10] form to describe the (indices) of the source and destination arguments involved in the attack

[9] Considering other possible choices of formal tools for the specification of infinite structures is beyond the scope of the present paper and is left for future work.

[10] We could, of course, use an arbitrary number base, however, to do so adds nothing in the way of expressive power and can, in fact, reduce this considerably.

with the symbol 1 used to separate these two components (in the absence of any attack $L_{\mathcal{A}} = \emptyset$).

For naive encodings one can consider formal grammars and their associated languages as a means of presenting a finite specification of $\langle \mathcal{X}, \mathcal{A} \rangle$, i.e. as a grammar G over alphabet $\{0, 1\}$ for which $L(G) = L_{\mathcal{A}}$. We show (proofs are given in Appendix B.1) that different choices for the family of grammars G belongs to lead invariably to the violation of (at least) one of the three criteria above, making this approach unsuitable in spite of its apparent simplicity.

We start with an unsurprising property of naive encodings.

Proposition 1. For $\mathcal{X}$ as introduced in Definition 10, there are choices of $\mathcal{A} \subseteq \mathcal{X} \times \mathcal{X}$ such that there is no formal grammar, G with $L(G) = L_{\mathcal{A}} \subseteq \{0^i \cdot 1 \cdot 0^j \ : \ i, \ j \geq 1\}$.

The issue of infinite attack structures which cannot be described within naive encodings may, justifiably, be seen as a purely technical limitation as far as the families of AFs so affected are unlikely to feature in applications (see the proof of Proposition 1). It turns out however that, for unrestricted grammars, the criterion of tractability is not satisfied since the problem of determining if a naive encoding does indeed describe *some* AF is not semi-decidable.

Proposition 2. Given an arbitrary (i.e. unrestricted) grammar G over the alphabet $\{0, 1\}$ the problem of determining if $L(G) \subseteq \{0^i \cdot 1 \cdot 0^j \ : \ i, \ j \geq 1\}$ is not semi-decidable, i.e. there is no TM program which given (a description of) G as input halts and accepts precisely those G for which $L(G) \subseteq \{0^i \cdot 1 \cdot 0^j \ : \ i, \ j \geq 1\}$.

In fact, results analogous to Proposition 2 continue to hold even if we use the less expressive class of context-sensitive grammars.

Proposition 3. Given an arbitrary *context-sensitive* grammar, G, over the alphabet $\{0, 1\}$ the problem of determining if $L(G) \subseteq \{0^i \cdot 1 \cdot 0^j \ : \ i, \ j \geq 1\}$ is not semi-decidable.

The problem evidenced in Propositions 2 and 3 does not hold when considering context-free languages.

Proposition 4. Given an arbitrary *context-free* grammar (CFG), G, over the alphabet $\{0, 1\}$ the problem of determining if $L(G) \subseteq \{0^i \cdot 1 \cdot 0^j \ : \ i, \ j \geq 1\}$ is decidable.

Context-free languages, however, do not satisfy the property of closure: it is well-known that they are not closed under intersection, complement and set difference.

Turning to regular languages, as with the context-free case one can decide if a given DFA accepts the naive encoding of some $\langle \mathcal{X}, \mathcal{A} \rangle$.

Proposition 5. Given $M = \langle Q, \{0, 1\}, q_0, F, \delta \rangle$ a DFA over the alphabet $\{0, 1\}$ there is a polynomial (in $|Q|$) algorithm that decides $L(M) \subseteq \{0^i \cdot 1 \cdot 0^j \ : \ i, \ j \geq 1\}$.

Moreover, regular languages are fully satisfactory as far as closure properties are concerned (see Appendix A Fact 6). Unfortunately, however, they feature very limited expressive power in terms of describing naive encodings.

Proposition 6. Let L be any subset of $\{0^i \cdot 1 \cdot 0^j \ : \ i, \ j \geq 1\}$ with the following property: there are infinitely many values of k such that $\{0^k \cdot 1 \cdot 0^m \ : \ m \geq 1\} \cap L \neq \emptyset$ *and* for all $0^n \cdot 1 \cdot 0^m \in L$, $n \leq m$. Then L is *not* a regular language.

Notice that one consequence of Proposition 6 is that the naive encoding of the infinite AF whose only attacks are *self-attacks*, i.e. $\langle p, p \rangle$ for all $p \in \mathcal{X}$ fails to be a regular language.[11]

In summary, although naive representations have an appealing structural simplicity, if adopted one has to contend with issues of undecidability (for the most expressive grammar classes), lack of closure (for context-free languages) or limited expressiveness (for regular languages).

5. A Generic Regular Expression Formalism and its Properties

The issues identified with so-called naive representations in the preceding section largely stem from the following fact: given that encodings of *arguments*, $p_i \in \mathcal{X}$ are effectively achieved for free – that is, for all natural numbers k, $p_k \in \mathcal{X}$ and no further analysis is needed – the task of describing $\langle \mathcal{X}, \mathcal{A} \rangle$ comes down to describing the (infinite) set $\mathcal{A}$. In assuming that $p_k \in \mathcal{X}$ for any k, however, this severely limits the extent to which $\mathcal{A}$ can be described in a computationally useful manner.

In this section we present an alternative method for describing infinite AFs, $\langle \mathcal{X}, \mathcal{A} \rangle$. The basic idea is that, rather than assuming $\mathcal{X}$ is understood simply as $\{p_1, p_2, \ldots, p_n, \ldots\}$, we consider arguments in $\mathcal{X}$ to be specified so that there is some structural aspect linking them. In this way we can then present very general specifications of the attack structure that are conditioned solely in terms of the *specific* arguments in $\mathcal{X}$.

In a nutshell, the proposal consists of two basic elements: a description of the (infinite) set of arguments through an appropriate *argument encoding* relying on some finite automaton, and a description of the attack relations linking arguments together through an *attack expression*. More precisely, the attack expression specifies a mapping between regular expressions describing sets of arguments, with the intended meaning that the set S is attacked by the elements of the set obtained from S through the mapping. The combination of the automaton describing the set of arguments and of the attack expression will be called the AF *specification*.

Example 3. To illustrate our approach we will use some "simple" infinite structures, which can be regarded as basic patterns, possibly to be reused in the context of more articulated infinite argumentation frameworks. A first example of

[11] This language, $\{0^m \cdot 1 \cdot 0^m \ : \ m \geq 1\}$ is, however, context-free.

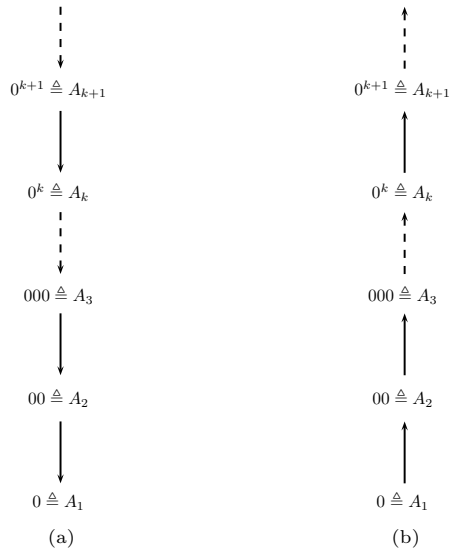

Figure 1: Two infinite attack chains: AF_D (a), and AF_U (b).

basic structure (related to endless debates, or "chiken and egg" style dilemmas, or the kicker and goalkeeper example) is an infinite sequence of arguments linked by the attack relation. We note that this kind of structure admits two different instantiations: one where the sequence "starts" with an attacked argument (which corresponds to the examples mentioned above) and a dual one where the sequence "starts" with an attacking argument. They correspond respectively to the argumentation frameworks AF_D and AF_U (see Figure 1) defined as follows: $AF_D = \langle \mathcal{X}, \mathcal{A}_D \rangle$ with $\mathcal{X} = \{A_1, A_2, \ldots, A_n, \ldots\}$ and $\mathcal{A}_D = \{\langle A_{i+1}, A_i \rangle \ : \ i \geq 1\}$ and $AF_U = \langle \mathcal{X}, \mathcal{A}_U \rangle$ with $\mathcal{X}$ as above and $\mathcal{A}_U = \{\langle A_i, A_{i+1} \rangle \ : \ i \geq 1\}$. Another example of basic structure, related to temporal projection, is a couple of "parallel" sequences of arguments where corresponding arguments in the sequences mutually attack each other. This kind of structure may correspond to conflicting information acquired at the same time for the same entity (e.g. a physical quantity) from two different and equally reliable sources A and B (e.g. two different experiments or measurements). The conflict between the initial arguments corresponding to the readings from the two sources is then projected over time. This can be represented by the argumentation framework $AF_M = \langle \mathcal{X}_M, \mathcal{A}_M \rangle$ with $\mathcal{X}_M = \{A_1, A_2, \ldots, A_n, \ldots\} \cup \{B_1, B_2, \ldots, B_n, \ldots\}$ and $\mathcal{A}_M = \{\langle A_i, B_i \rangle \ : \ i \geq 1\} \cup \{\langle B_i, A_i \rangle \ : \ i \geq 1\}$ (see Figure 2(a)). To provide a slightly more complicated structure, we will consider also a variant of this temporal projection situation, with a third source C which is in agreement with the source A and is considered more reliable than A and B. This can be represented by the argumentation framework $AF_R = \langle \mathcal{X}_R, \mathcal{A}_R \rangle$ with $\mathcal{X}_R = \{A_1, A_2, \ldots, A_n, \ldots\} \cup \{B_1, B_2, \ldots, B_n, \ldots\} \cup \{C_1, C_2, \ldots, C_n, \ldots\}$ and

$\mathcal{A}_R = \{\langle A_i, B_i\rangle \ : \ i \geq 1\} \cup \{\langle B_i, A_i\rangle \ : \ i \geq 1\} \cup \{\langle C_i, B_i\rangle \ : \ i \geq 1\}$ (see Figure 2(b)).

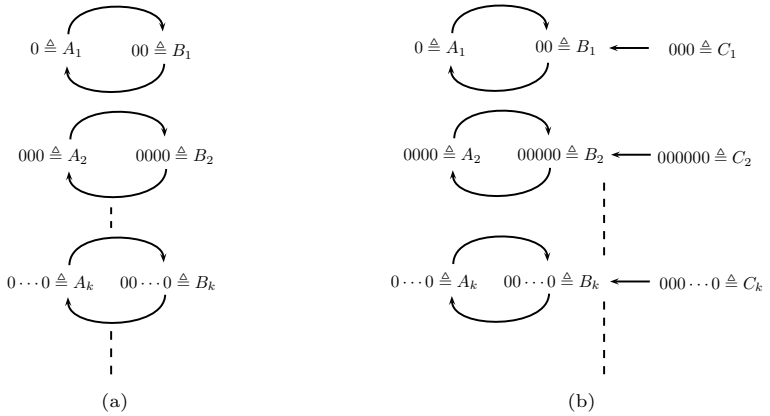

Figure 2: Two infinite argumentation frameworks: AF_M (a), and AF_R (b).

5.1. Argument encoding

Central to our formalism is the notion of "argument encoding" as a given set of words over some base set of symbols.

Definition 11. Let $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ be an alphabet. An *argument encoding* over Σ is any regular language $\mathcal{X} \subseteq \Sigma^* \setminus \{\varepsilon\}$.

Thus a (possibly infinite) set of arguments, $\mathcal{X}$, corresponds to some regular language over a finite alphabet Σ.

Example 3 (continued). The notion of argument encoding is very general and, in fact, leaves completely open the choice of the alphabet and of the regular language to be adopted to represent a given infinite structure. Considering the frameworks AF_D and AF_U, featuring a simple "linear" structure, the straightforward choice we will follow uses an alphabet consisting of a unique symbol, actually $\Sigma = \{0\}$, and the regular language $0 \cdot 0^*$, by adopting the correspondence $0^i \triangleq A_i$. For frameworks with more articulated structures like AF_M and AF_R, one might consider using an alphabet with a symbol for each class of arguments, e.g. $\Sigma_M = \{0, 1\}$ and $\Sigma_R = \{0, 1, 2\}$. Accordingly the regular languages to encode $\mathcal{X}_M$ and $\mathcal{X}_R$ would be $0 \cdot 0^* \cup 1 \cdot 1^*$ and $0 \cdot 0^* \cup 1 \cdot 1^* \cup 2 \cdot 2^*$ respectively, with the correspondences $0^i \triangleq A_i$, $1^i \triangleq B_i$, $2^i \triangleq C_i$. As we will see later, however, a different approach will be adopted since it is more suitable for the purpose of representation of the attack relation. In fact, we will use again the alphabet $\Sigma = \{0\}$, and the regular language $0 \cdot 0^*$ for all arguments, putting the different classes of arguments in correspondence with distinct sublanguages

of $0 \cdot 0^*$. In fact for AF_M we will adopt the correspondence $0 \cdot (00)^i \triangleq A_{i+1}$ and $00 \cdot (00)^i \triangleq B_{i+1}$ for $i \geq 0$, i.e. $0 \triangleq A_1$, $00 \triangleq B_1$, $000 \triangleq A_2$, $0000 \triangleq B_2$ and so on. Similarly for AF_R we will adopt the correspondence $0 \cdot (000)^i \triangleq A_{i+1}$, $00 \cdot (000)^i \triangleq B_{i+1}$ and $000 \cdot (000)^i \triangleq C_{i+1}$for $i \geq 0$.

5.2. Attack expression

The increased expressivity and computational effectiveness of our approach (in comparison to naive encodings), derives from the mechanism used to describe sets of attacks. We seek to develop formalisms by which the set of arguments (in $\mathcal{X}$) that "attack" in the standard sense of [37] some specified subset $S \subseteq \mathcal{X}$ may be presented, i.e. for defining "suitable" functions $\mu : 2^{\mathcal{X}} \rightarrow 2^{\mathcal{X}}$, such that, for a given set of arguments S, $\mu(S)$ specifies the set of arguments attacking S, i.e. $\mu(S) = S^-$. By "suitable" we recognise that there are certain natural conditions that such functions ought to respect.

Definition 12. A mapping $\mu : 2^{\Sigma^*} \rightarrow 2^{\Sigma^*}$ is *reasonable* if

R1. $\forall S \subseteq \Sigma^*$, $\mu(S) = \bigcup_{u \in S} \mu(\{u\})$. (Additivity)
R2. $\forall S \subseteq \Sigma^*$ for which S is a regular language, $\mu(S)$ is a regular language. (Closure)

In addition, we say that a reasonable mapping is *invertible* if the function $\nu : 2^{\Sigma^*} \rightarrow 2^{\Sigma^*}$ defined via

$$\nu(S) = \{u \in \Sigma^* : \exists\, v \in S \text{ s.t. } v \in \mu(\{u\})\}$$

is in turn a reasonable mapping. It is easy to see that ν satisfies property R1 by definition.

We observe that all of these conditions hold for the finite instantiations of $\mathcal{A}$ in Dung's AFs. For the development we consider to infinite $\mathcal{X}$, the additivity restriction states that attacks on S must be associated with *individual* arguments in S and not with S as a whole, i.e. we can construct the set of attacks on S simply by considering attacks on the members of S in turn.[12] It may be noted that additivity implies that the mapping is also monotonic, i.e. if $R \subseteq S$ then $\mu(R) \subseteq \mu(S)$. Thus additive mappings reflect the natural condition that attacks on a set of arguments, S, cannot be eliminated simply by adding more arguments to S: notice that by disallowing *explicit removal* of an attack on a set (in the definition of μ) we require *defences* to attacks to be made by direct counterattacks. In this way we preserve the concept of "u is acceptable to S" by identifying any $v \in S$ such that $v \in \mu(\{x\})$ for each $x \in \mu(\{u\})$.

While the additivity may be justified through semantic considerations, our reason for imposing "closure wrt regular languages" is motivated by computational concerns: given that $S \subseteq \mathcal{X}$ (if regular) has a simple computational

[12] We note, however, that settings with attack relations not respecting additivity have been examined in the context of finite frameworks, e.g. Nielsen and Parsons [69] and Bochmann [20].

representation (e.g. as a DFA accepting exactly the arguments in S) it is desirable that subsets of $\mathcal{X}$ "related to" S, e.g. through the property of attacking its members, can also be so described. In principle this allows for the outcome of $\mu(S)$ to be described as a DFA.

Finally the concept of invertibility addresses issues arising with the "inverse" mapping: just as $\mu : 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ describes the subset of $\mathcal{X}$ that *attacks* a given $S \subseteq \mathcal{X}$, so $\nu : 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ describes the subset of $\mathcal{X}$ that is *attacked by* S. The justification of additivity and closure properties is on similar grounds to those used with μ.

Given the basic desiderata that mappings defining attacks ought to satisfy stated in Definition 12, we now turn to the issue of how general functions satisfying these desiderata can be constructed. For this purpose we introduce the concept of attack expressions over a finite alphabet Σ.

Definition 13. A well-formed *attack expression* (AE) over Σ is a sentence constructed by the following rules.

1. For all $\sigma_i \in \Sigma$, σ_i is an attack expression over Σ.
2. The symbol I (for *identity*) is an attack expression over Σ.
3. If p and q are two attack expressions over Σ then $p \cup q$ is also an attack expression over Σ.
4. If p is an attack expression and K_Σ is a regular expression (using the operations $\{+, \cdot, *\}$) over *only* symbols from Σ (i.e. the identity symbol I does *not* occur in K_Σ) then all of $K_\Sigma \cdot p$, $p \cdot K_\Sigma$, p/K_Σ, K_Σ/p and $p \cap K_\Sigma$ are attack expressions.
5. If p is an attack expression over Σ then (p) and $\gamma(p)$ for $\gamma \in \{hd,\ tl\}$ are attack expressions over Σ.
6. The only attack expressions over Σ are those formed by a finite number of applications of (1) through (5).

Let $\mathcal{AE}(\Sigma)$ denote the set of all well-formed attack expressions over Σ and for $a \in \mathcal{AE}(\Sigma)$ let $size(a)$ denote the number of *operations* (i.e. applications of rules 3–5) used to define a. The key motivation for this formalism is in describing the *attack* structure relating a set of arguments. Hence each $a \in \mathcal{AE}(\Sigma)$, defines a mapping $\underline{a} : 2^{\Sigma^*} \to 2^{\Sigma^*}$ as follows

Definition 14. Let $p \in \mathcal{AE}(\Sigma)$ and $S \subseteq \Sigma^*$, the set $\underline{p}(S)$ is given by the rules below:

$$\underline{p}(S) = \begin{cases} \{\sigma_i\} & \text{if } p = \sigma_i \\ S & \text{if } p = I \\ (\underline{b}(S) \cup \underline{c}(S)) & \text{if } p = (b \cup c) \\ K_\Sigma \cdot \underline{b}(S) & \text{if } p = K_\Sigma \cdot b \\ \underline{b}(S) \cdot K_\Sigma & \text{if } p = b \cdot K_\Sigma \\ \underline{b}(S)/K_\Sigma & \text{if } p = b/K_\Sigma \\ K_\Sigma/\underline{b}(S) & \text{if } p = K_\Sigma/b \\ \underline{b}(S) \cap K_\Sigma & \text{if } p = b \cap K_\Sigma \\ \gamma(\underline{b}(S)) & \text{if } p = \gamma(b) \end{cases}$$

The unary operations – $\{hd,\ tl\}$ – are defined as follows for $S \subseteq \Sigma^*$:

$$hd(S) \ = \ \{\ \sigma_i\ \in \Sigma :\ \exists w \text{ s.t. } \sigma_i \cdot w \in S\}$$

$$tl(S) \ = \ \{\ w\ :\ \exists\ \sigma_i\ \in \Sigma \text{ s.t. } \sigma_i \cdot w \in S\}$$

Note that $tl(\{\sigma\}) = \{\varepsilon\}$ and $tl(\emptyset) = hd(\{\varepsilon\}) = hd(\emptyset) = \emptyset$.

Before analysing the properties of the proposed attack expressions, we comment on the set of operations – $\{\cup,\ \cdot K_\Sigma,\ K_\Sigma \cdot,\ /K_\Sigma,\ K_\Sigma /,\ \cap K_\Sigma,\ hd,\ tl\}$ that are provided. In particular we note the limited way in which $\cdot$ and $\cap$ may be used and the absence of complement and Kleene $*$ operators, despite the fact that regular languages are closed under the last two of these. The immediate problem with allowing arbitrary usage of $\cdot$ and $\cap$ concerns the fact that, for expressions such as $p \cdot q$ or $p \cap q$ we cannot, in general, guarantee that the mappings $\underline{p} \cdot \underline{q}$ or $\underline{p} \cap \underline{q}$ are *additive*. For example, suppose that $p = q = I$, $\Sigma = \{0, 1\}$ and $S = \{0^k\ :\ k \geq 1\}$. Then $(\underline{p} \cdot \underline{q})(S) = S \setminus \{0\}$, however

$$\bigcup_{w \in S} (\underline{p} \cdot \underline{q})(\{w\}) \ = \ \{0^{2k}\ :\ k \geq 1\} \quad \neq S \setminus \{0\}$$

Similarly if $p = I$, $q = tl(I)$, $S = \{1^k\ :\ k \geq 1\} \cup \{0 \cdot 1^k\ :\ k \geq 1\}$ we get

$$(\underline{p} \cap \underline{q})(S) \ = \ S \cap tl(S) \ = \{1^k\ :\ k \geq 1\}$$

but

$$\bigcup_{w \in S} (\underline{p} \cap \underline{q})(\{w\}) \ = \ \bigcup_{w \in S} \underline{p}(\{w\}) \cap \underline{q}(\{w\}) \ = \ \bigcup_{w \in S} (\{w\} \cap \{tl(w)\}) \ = \ \emptyset$$

We could, of course, avoid this by directly defining $\mu(S)$ to be $\bigcup_{w \in S}\ \mu(\{w\})$, i.e. restricting the domain of μ to Σ^*. In this case, however, we cannot always ensure that $\underline{p} \cdot \underline{q}$ preserves regularity. For example, using $p = I$, $q = \{1\} \cdot I$ with $S = \{0^k\ :\ k \geq 1\}$ (which is a regular language),

$$\bigcup_{w \in S} \underline{p}(w) \cdot \underline{q}(w) \ = \ \{0^k \cdot 1 \cdot 0^k\ :\ k \geq 1\}$$

is not regular. This behaviour arises since regular languages are not closed under *unbounded* union (only with respect to *finite* union). The issues underpinning the absence of $\cdot$ from the class of allowed operations are easily seen also to arise were $*$ to be added. Finally allowing complementation would lead to mappings which were not monotonic (and thus could not be additive).

Theorem 1 provides a first confirmation of the soundness of the proposed approach by showing[13] that the mappings arising via attack expressions have appropriate properties.

Theorem 1. Let $p \in \mathcal{AE}(\Sigma)$ be *any* attack expression over Σ. The mapping $\underline{p}\ :\ 2^{\Sigma^*} \to 2^{\Sigma^*}$ is reasonable.

[13]The proofs relevant to this section are given in Appendix B.2.

5.3. AF *specification*

We now have the basic elements of our formal descriptive mechanism for infinite frameworks, the idea being that the set of arguments is specified as a regular language $\mathcal{X} \subseteq \Sigma^*$ and the attack relation is specified through an attack expression a. In fact, given an element v of $\mathcal{X}$ the set T of attackers of v might be defined as $T = \underline{a}(\{v\})$. Considering now a set $S \subseteq \mathcal{X}$, the set of attackers of S is given by $\{v \in \Sigma^* \ : \ \exists\ u \in S \text{ s.t. } v \in \underline{a}(\{u\})\}$ which is equal to $\underline{a}(S)$ by additivity of $\underline{a}$. However, it should be noted that while the attack relation must be a subset of $\mathcal{X} \times \mathcal{X}$, it might be the case that for some $S \subseteq \mathcal{X}$ $\underline{a}(S) \not\subseteq \mathcal{X}$. To fix this (actually minor) problem we need to introduce some further notation.

Definition 15. Let $\mu \ : \ 2^{\Sigma^*} \to 2^{\Sigma^*}$ and $\mathcal{X} \subseteq \Sigma^*$ a regular language, we define $\mu_{\mathcal{X}} : \ 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ as

$$\mu_{\mathcal{X}}(S) \quad = \quad \mu(S) \cap \mathcal{X}$$

Proposition 7 shows that considering $\mu_{\mathcal{X}}$ instead of μ does not affect the property of being a reasonable mapping.

Proposition 7. If μ is a reasonable mapping over the domain 2^{Σ^*} then $\mu_{\mathcal{X}}$ is a reasonable mapping over the domain $2^{\mathcal{X}}$.

We can now formally introduce the notion of AF specification.

Definition 16. Let Σ be a finite alphabet of symbols and $\mathcal{X} \subseteq \Sigma^*$ be a regular language. An AF *specification* (AFS) is a pair $\langle \mathcal{M}, a \rangle$ where $\mathcal{M} = \langle Q, \Sigma, q_0, \delta, F \rangle$ is some finite automaton[14] for which $L(\mathcal{M}) = \mathcal{X}$ and $a \in \mathcal{AE}(\Sigma)$ is a well-formed attack expression over Σ. Given an AFS $\langle \mathcal{M}, a \rangle$, the relation $\rightarrow_a$ over $\mathcal{X} \times \mathcal{X}$ is defined by $u \rightarrow_a v$ (read as "v is attacked by u") if $u \in \underline{a}_{\mathcal{X}}(\{v\})$. We call $\langle \mathcal{X}, \rightarrow_a \rangle$ the argumentation framework induced by $\langle \mathcal{M}, a \rangle$.

Note that, given an AF *specification* and a set $S \subseteq \mathcal{X}$, the set of attackers of S denoted as $\pi_a^-(S) \triangleq \{\ v \in \Sigma^* \ : \ \exists\ u \in S \text{ s.t. } v \rightarrow_a u\}$ is equal to $\underline{a}_{\mathcal{X}}(S)$, by additivity.

Example 3 (continued). Continuing with the examples, we can now complete the specification of AF_U, AF_D, AF_M and AF_R by identifying the relevant attack expressions.

As to AF_U we note that, for a generic $i \geq 2$ the argument (corresponding to) 0^i is attacked by the argument 0^{i-1}, leading to the attack expression $a = tl(I)$. Note that $\underline{a}(\{0\}) = \{\varepsilon\}$ but $\underline{a}_{\mathcal{X}}(\{0\}) = \emptyset$. On the other hand, in AF_D, for a generic $i \geq 1$, the argument 0^i is attacked by the argument 0^{i+1}, leading to the attack expression $I \cdot 0$.[15]

AF_M requires a more articulated attack expression. Here each argument A_i corresponding to $0 \cdot (00)^{i-1}$ is attacked by the argument B_i corresponding to

[14] We do not require that $\mathcal{M}$ be limited to a *specific* class of automata since there is no expressive gain in imposing such a restriction (see Fact 5 in Appendix A).

[15] Of course, we could equally write $0 \cdot I$ in this case.

$00 \cdot (00)^{i-1}$ and viceversa each argument B_i is attacked by the argument A_i. As to the attacks from a generic B_i to a generic A_i we note that the attacker can simply be obtained by adding a trailing 0. We can not however use the simple expression $I \cdot 0$ as above since this would entail not only that any A_i is attacked by B_i but also that any B_i is attacked by A_{i+1} which is not the case. The attack expression has therefore to specify that the trailing 0 applies only to the elements of the sublanguage $0 \cdot (00)^*$, giving rise to the expression $(I \cap (0 \cdot (00)^*)) \cdot 0$. Similarly, the attacks from a generic A_i to a generic B_i can be obtained using the tl operator and properly restricting its application, giving rise to $tl(I \cap ((00) \cdot (00)^*))$. The complete attack expression of AF_M is obtained by the union of the two expressions above: $a_M = ((I \cap (0 \cdot (00)^*)) \cdot 0) \cup tl(I \cap ((00) \cdot (00)^*))$.

As to AF_R, we note first that the attacks between arguments A_i and B_i are analogous to the case of AF_M with the difference that A_i corresponds to $0 \cdot (000)^{i-1}$ and B_i corresponds to $00 \cdot (000)^{i-1}$. Hence, similarly to above, we obtain the expressions $(I \cap (0 \cdot (000)^*)) \cdot 0$ and $tl(I \cap ((00) \cdot (000)^*))$. As to the attacks from an argument C_i, corresponding to $000 \cdot (000)^{i-1}$, to an argument B_i corresponding to $00 \cdot (000)^{i-1}$ we note that they can again be represented through the addition of a trailing 0 yielding $(I \cap (00 \cdot (000)^*)) \cdot 0$. The complete attack expression of AF_R turns out to be $a_R = ((I \cap (0 \cdot (000)^*)) \cdot 0) \cup (tl(I \cap ((00) \cdot (000)^*))) \cup ((I \cap (00 \cdot (000)^*)) \cdot 0)$.

As a further remark, we note that the attack expression I captures the case of an argumentation framework where each argument attacks itself (and only itself) which has been shown not to be representable in the naive approach using regular languages in Section 4.

5.4. Inverting the attack expression

Definition 16 provides a formal specification of the attackers of an element (subset) of Σ^*. However it is useful to consider also the specification of the arguments attacked by a given element (subset) of Σ^*. This is possible since the function $\underline{a}$ is invertible as shown below by Theorem 2. In particular, the proof of the theorem (see Appendix B.2) allows one to construct from a given $a \in \mathcal{AE}(\Sigma)$ a related expression a^+ with the property that for all $u,\ v \in \Sigma^*$, $u \in \underline{a}(\{v\})$ if and only if $v \in \underline{a}^+(\{u\})$. The expressions a^+ use the same basic elements as $\mathcal{AE}(\Sigma)$ plus the $rev()$ operator[16], which does not affect any of the desired properties.

The following properties, shown to be valid in the proof of Theorem 2, provide the basic elements to derive the expressions a^+ from $\underline{a}$.

Fact 2.

1. If $a = \sigma_i$ then $\underline{a}^+(S) = tl(I \cap \sigma_i) \cdot \Sigma^*$; If $a = I$ then $\underline{a}^+(S) = S$.
2. If $a = b \cup c$, then $\underline{a}^+(S) = \underline{b}^+(S) \cup \underline{c}^+(S)$.

[16] With a little abuse of notation, in order to simplify the presentation, we will apply the $rev()$ operator also to single words, i.e. for $w \in \Sigma^*$ and $\{w'\} = rev(\{w\})$, $rev(w) = w'$.

3. If $a = b \cdot K_\Sigma$, then $\underline{a}^+(S) = \underline{b}^+(S/K_\Sigma)$.
4. If $a = K_\Sigma \cdot b$, then $\underline{a}^+(S) = \underline{b}^+(rev(rev(S)/rev(K_\Sigma)))$.
5. If $a = b/K_\Sigma$, then $\underline{a}^+(S) = \underline{b}^+(S \cdot K_\Sigma)$.
6. If $a = K_\Sigma/b$, then $\underline{a}^+(S) = \underline{b}^+(rev(rev(K_\Sigma)/rev(S)))$.
7. If $a = b \cap K_\Sigma$, then $\underline{a}^+(S) = \underline{b}^+(S \cap K_\Sigma)$.
8. If $a = hd(b)$, then $\underline{a}^+(S) = \underline{b}^+(((S \cap \Sigma) \cdot \Sigma^*))$.
9. If $a = tl(b)$, then $\underline{a}^+(S) = \underline{b}^+(\Sigma \cdot S)$.

Theorem 2. Let $\langle \mathcal{M}, a \rangle$ be an AFS with $L(\mathcal{M}) = \mathcal{X}$ and $a \in \mathcal{AE}(\Sigma)$. The mapping $\underline{a}^+ : 2^{\Sigma^*} \to 2^{\Sigma^*}$, defined as $\underline{a}^+(S) = \{ v \in \Sigma^* \ : \ \exists\, u \in S \text{ s.t. } u \in \underline{a}(v)\}$ is closed wrt regular languages.

It remains now to show that the inverse of an attack expression actually provides the set of the arguments attacked by a set.

Proposition 8. Let $\langle \mathcal{M}, a \rangle$ be an AFS with $L(\mathcal{M}) = \mathcal{X}$ and $a \in \mathcal{AE}(\Sigma)$. Define the mapping, $\pi_a^+ \ : \ 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ by

$$\pi_a^+(S) = \{ v \in \mathcal{X} \ : \ \exists\, u \in S \text{ s.t. } u \to_a v\}$$

It holds that $\pi_a^+(S) = \underline{a}_{\mathcal{X}}^+(S)$.

By Theorem 2 and the regularity of $\mathcal{X}$ it is easy to see that π_a^+ is additive and preserves regularity.

Example 3 (continued). Completing our example, we can now derive the mapping π_a^+ for AF_U, AF_D, AF_M and AF_R.

As to AF_U from the attack expression $tl(I)$ applying Fact 2.9 (and 2.1 for I) we get $\underline{a}^+(S) = I(\Sigma \cdot S) = \Sigma \cdot S = 0 \cdot S$. As to AF_D, from the attack expression $I \cdot 0$ applying Fact 2.3 (and 2.1 for I) we get $\underline{a}^+(S) = I(S/0) = S/0$ which (in view of $S \subseteq 0 \cdot 0^*$) is equivalent to $tl(S)$.

As to AF_M, given the attack expression $a_M = ((I \cap (0 \cdot (00)^*)) \cdot 0) \cup tl(I \cap ((00) \cdot (00)^*))$ by Fact 2.2 we can examine separately the two terms $b = ((I \cap (0 \cdot (00)^*)) \cdot 0)$ and $c = tl(I \cap ((00) \cdot (00)^*))$. As to $\underline{b}^+$, from 2.3 we get $\underline{d}^+(S/0)$ with $d = (I \cap (0 \cdot (00)^*))$. Applying then 2.7 (and 2.1 for I) we get $\underline{b}^+(S) = (S/0) \cap (0 \cdot (00)^*)$. As to $\underline{c}^+$, from Fact 2.9 we get $\underline{e}^+(\Sigma \cdot S)$ with $e = I \cap ((00) \cdot (00)^*)$. Applying again 2.7 and 2.1 and taking into account $\Sigma = \{0\}$ we get $\underline{c}^+(S) = (0 \cdot S) \cap ((00) \cdot (00)^*)$. Summing up $\underline{a_M}^+(S) = ((S/0) \cap (0 \cdot (00)^*)) \cup ((0 \cdot S) \cap ((00) \cdot (00)^*))$.

The case of AF_R is analogous, yielding $\underline{a_R}^+(S) = ((S/0) \cap (0 \cdot (000)^*)) \cup ((0 \cdot S) \cap ((00) \cdot (000)^*)) \cup ((S/0) \cap (00 \cdot (000)^*))$.

5.5. Representation of finite AFs *and combination of* AF *specifications*

The previous sections show how the proposed AF specification mechanism can deal with (up to now simple) infinite frameworks. One may then wonder whether this mechanism is suitable to describe *finite* AFs as well or its structure is somehow bounded to the infinite case. In fact this problem does not arise: any finite AF $\langle \mathcal{X}, \mathcal{A} \rangle$ can be easily described via the mechanisms proposed in this

paper. Noting that $\mathcal{X}$ is finite simply choose $\Sigma = \mathcal{X}$ as the underlying alphabet, and let $\mathcal{M}$ be the trivial associated automaton. The set $\mathcal{A}$ is a finite subset of $\Sigma \times \Sigma$ and treated directly as a regular language $L_A = \{x \cdot y : \langle x, y\rangle \in \mathcal{A}\}$. We then have $a \in \mathcal{AE}(\Sigma)$ specified by $a = hd((\Sigma \cdot I) \cap L_A)$, giving $\pi_a^-(S) = hd((\Sigma \cdot S) \cap L_A)$ and (after some manipulation) $\pi_a^+(S) = tl((S \cdot \Sigma) \cap L_A)$.

For example if $\mathcal{X} = \{w, x, y, z\}$ and $\mathcal{A} = \{\langle w, x\rangle, \langle x, y\rangle, \langle y, z\rangle, \langle z, w\rangle\}$ then $L_A = \{wx, xy, yz, zw\}$ with, for example,

$$\begin{aligned}
\pi_a^-(\{w, y\}) &= hd((\{w, x, y, z\} \cdot \{w, y\}) \cap \{wx, xy, yz, zw\}) \\
&= hd(\{ww, wy, xw, xy, yw, yy, zw, zy\} \cap \{wx, xy, yz, zw\}) \\
&= hd(\{xy, zw\}) \quad = \quad \{x, z\}
\end{aligned}$$

$$\begin{aligned}
\pi_a^+(\{w, y\}) &= tl((\{w, y\} \cdot \{w, x, y, z\}) \cap \{wx, xy, yz, zw\}) \\
&= tl(\{ww, yw, wx, yx, wy, yy, wz, yz\} \cap \{wx, xy, yz, zw\}) \\
&= tl(\{wx, yz\}) \quad = \quad \{x, z\}
\end{aligned}$$

Having shown that finite AFs do not raise, *per se* any expressiveness concern, a further important question has to be addressed: one may wonder whether it is possible to give the specification of an AF resulting from the combination of a finite subframework with one or more infinite subframeworks, with the different subframeworks linked together by finite attack relations. This kind of combined specification is particularly relevant in practice. To have an example, consider again the frameworks AF_M and AF_R, concerning cases of temporal projection with initial information acquired from different sources at the same time. Clearly, one has also to cover the case where information is acquired from different sources at different times. As a very simple example, consider a slight modification of the situation represented by AF_R, so that information from the third more reliable source, C, is acquired with some delay (to keep things simple, let say one time instant later) wrt the information from sources A and B. This situation could be represented with a framework composed by two subframeworks, a finite one, consisting of two mutually attacking arguments corresponding to the information initially acquired from A and B, and an infinite one, with the same structure as AF_R. More generally, frameworks with this kind of structure correspond to cases where a reasoning (or dialogue) process enters a non terminating iterative behavior after some initial non iterative steps, which is clearly a more general (and possibly more common) situation wrt the cases of "iterative behavior from the beginning" we have considered in our simple illustrative examples. We will now show how this kind of structure can be captured in our formalism.

Let $AF_0 = \langle \mathcal{X}_0, \mathcal{A}_0\rangle$ a finite AF and $AF_1 = \langle \mathcal{X}_1, \mathcal{A}_1\rangle, \ldots, AF_n = \langle \mathcal{X}_n, \mathcal{A}_n\rangle$ a finite sequence of infinite frameworks with specifications $\langle \mathcal{M}_1, a_1\rangle, \ldots, \langle \mathcal{M}_n, a_n\rangle$ such that for each $1 \leq i \leq n$ $L(\mathcal{M}_i) = \mathcal{X}_i \subseteq \Sigma_i^*$ and $a_i \in \mathcal{AE}(\Sigma_i)$.

Letting $\Sigma_0 = \mathcal{X}_0$ we assume without loss of generality that the alphabets used for the different frameworks are pairwise disjoint namely, for $0 \leq i \leq n$, $0 \leq j \leq n$, $i \neq j$, $\Sigma_i \cap \Sigma_j = \emptyset$.

Consider now the problem of specifying a framework $AF_\cup = \langle \mathcal{X}_\cup, \mathcal{A}_\cup\rangle$ with the following structure:

- $\mathcal{X}_\cup = \bigcup_{i\in\{0,\ldots,n\}} \mathcal{X}_i$;
- $\mathcal{A}_\cup = \bigcup_{i\in\{0,\ldots,n\}} \mathcal{A}_i \cup \bigcup_{i,j\in\{0,\ldots,n\}, i\neq j} \mathcal{A}_{i,j}$, where $\mathcal{A}_{i,j}$ is an arbitrary *finite* subset of $\mathcal{X}_i \times \mathcal{X}_j$.

In words, $AF_\cup$ includes all the subframeworks $AF_0, \ldots AF_n$ with their "internal" attack relations $\mathcal{A}_0, \ldots \mathcal{A}_n$ plus new arbitrary *finite* attack relations $\mathcal{A}_{i,j}$ linking each pair of subframeworks and representing the additional attacks from elements of $\mathcal{X}_i$ to elements of $\mathcal{X}_j$.

The question is now how to derive the specification of $AF_\cup$ from the specifications of the subframeworks $AF_0, \ldots AF_n$ and from the new attacks $\mathcal{A}_{i,j}$. As to the reference alphabet, clearly $\Sigma_\cup = \bigcup_{i\in\{0,\ldots,n\}} \Sigma_i$. Thanks to the hypothesis of disjointness of the alphabets Σ_i, we know that also the sets of arguments $\mathcal{X}_i$ are disjoint and we can safely define $\mathcal{X}_\cup = \bigcup_{i\in\{0,\ldots,n\}} \mathcal{X}_i$. As $\mathcal{X}_\cup$ is the union of a set of regular languages it is a regular language too (Fact 4, Appendix A), whose automaton $\mathcal{M}_\cup$ can be effectively derived from the automata $\mathcal{M}_i$ (Fact 6.b, Appendix A).

As to the attack expression, it has to preserve, in the new framework, the attack relations $\mathcal{A}_0, \ldots \mathcal{A}_n$ of all the subframeworks and include the new attacks $\mathcal{A}_{i,j}$.

As to $\mathcal{A}_0$, the corresponding attack expression in $AF_\cup$ is exactly the same as for $\mathcal{A}_0$ in isolation. Letting $L_0 = \{x \cdot y : \langle x, y\rangle \in \mathcal{A}_0\}$, the attack expression[17] for the finite subframework is $\hat{a}_0 = hd((\Sigma_0 \cdot I) \cap L_0)$.

As to the attack relations $\mathcal{A}_1, \ldots \mathcal{A}_n$ of the infinite subframeworks $AF_1, \ldots AF_n$, each relation $\mathcal{A}_i$ is described by an attack expression a_i and we need to devise a corresponding attack expression $\hat{a}_i$ which preserves exactly the same attacks between the arguments of AF_i in the context of $AF_\cup$, i.e. for any set $S \subseteq \Sigma_\cup$ it must hold that $\underline{\hat{a}}_i(S) = \underline{\hat{a}}_i(S \cap \mathcal{X}_i) = \underline{a}_i(S \cap \mathcal{X}_i)$. It can be seen that such an expression $\hat{a}_i$ can be obtained from a_i by applying two simple replacement operations concerning the basic elements σ and I (Rules 1 and 2 of Definition 13):

- each occurrence of σ (with σ an element of Σ_i) in a_i is replaced by $hd(\sigma \cdot (I \cap K_i))$ within $\hat{a}_i$, where K_i is the regular expression specifying $\mathcal{X}_i$;
- each occurrence of I in a_i is replaced by $(I \cap K_i)$ within $\hat{a}_i$, where K_i is as above.

It is immediate to see that if the attack expression a_i consists exactly of I or σ, the above replacements ensure that $\hat{a}_i$ satisfies the desired property. By inspection of the rules 3-5 of Definition 13 it is also easy to see that the desired property is preserved in more articulated expressions, constructed by repeated application of these rules starting from the basic elements, without requiring any further modification.

[17] We remark that using $\Sigma_\cup$ instead of Σ_0 would give the same result.

Let us turn now to the specification of the additional finite attack relations $\mathcal{A}_{i,j}$ $(i \neq j)$ between subframeworks. First, observe that each $\mathcal{A}_{i,j}$ can be specified directly as a regular language $L_{i,j} = \{x \cdot y : \langle x, y\rangle \in \mathcal{A}_{i,j}\}$ and let again K_i be the regular expression corresponding to $\mathcal{X}_i$. Then, note that the expression $(K_i \cdot I) \cap L_{i,j}$ is useful to select words in $L_{i,j}$ when appropriate and that the quotient operator wrt K_j can then be applied to extract the subword referring to the attacker. Finally, we have to ensure that the extracted subword belongs to K_i which can be obtained by an intersection operation. In summary, the attack expression specifying an attack relations $\mathcal{A}_{i,j}$ $(i \neq j)$ is $\hat{a}_{i,j} = (((K_i \cdot I) \cap L_{i,j})/K_j) \cap K_i$.

Putting together the various subexpressions we have devised above, the complete attack expression $a_\cup$ for $AF_\cup$ is given by $a_\cup = \bigcup_{i\in\{0,\ldots,n\}} \hat{a}_i \cup \bigcup_{i,j\in\{0,\ldots,n\}, i\neq j} \hat{a}_{i,j}$.

6. Computing with AF Specifications

In this section we show[18] that a number of problems that are well-known to be efficiently, i.e. polynomial time, decidable in finite AFs may be effectively handled within the context of AF specifications, i.e. there exist procedures which are certain to produce the answer in a finite number of steps.

Theorem 3 provides the main result of this section.

Theorem 3. Let $\langle \mathcal{M}, a\rangle$ be an AFS, with induced argumentation framework $\langle \mathcal{X}, \mathcal{A}\rangle$ and $S \subseteq \mathcal{X}$. The following problems are decidable.

a. Deciding if the set S is conflict free
b. For $x \in \mathcal{X}$, deciding if $x \in \mathcal{F}(S)$, i.e. whether x is acceptable to S
c. Deciding if $S \in \mathcal{E}_{adm}(\langle \mathcal{X}, \mathcal{A}\rangle)$, i.e. whether S is admissible
d. Deciding if $S \in \mathcal{E}_{stab}(\langle \mathcal{X}, \mathcal{A}\rangle)$, i.e. whether S is a stable extension
e. Constructing a DFA accepting $\mathcal{F}(S) = \mathcal{X} \setminus \pi_a^+(\mathcal{X} \setminus \pi_a^+(S))$, i.e. the set of arguments acceptable to S.
f. Deciding if $S \in \mathcal{E}_{comp}(\langle \mathcal{X}, \mathcal{A}\rangle)$, i.e. whether S is a complete extension

The algorithms described in the proof of Theorem 3 provide full solutions to the decision problems listed above and are applicable irrespective of whether $\langle \mathcal{M}, a\rangle$ gives rise to finitary frameworks or not. For the construction of the grounded extension we obtain a result applicable to finitary argumentation frameworks only. In fact, in [37] it is shown that, letting $\mathcal{F}^1(S) = \mathcal{F}(S)$, and for $i > 1$ $\mathcal{F}^i(S) = \mathcal{F}(\mathcal{F}^{i-1}(S))$, for a finitary argumentation framework the grounded extension is given by $\bigcup_{i\geq 1} \mathcal{F}^i(\emptyset)$.

Using this result we can in some cases obtain (a representation of) the grounded extension through Algorithm 1.

Algorithm 1 effectively reproduces the sequence of iterations involving:

- identifying unattacked arguments in $\mathcal{X}$ (in l.2 and l.7), where we note that Y_0 computes $\mathcal{F}(\emptyset)$, i.e. $\mathcal{X} \setminus \pi_a^+(\mathcal{X})$ since $\pi_a^+(\emptyset) = \emptyset$;

[18] The proofs relevant to this section are given in Appendix B.3.

Algorithm 1 Computing $\mathcal{E}_{gr}(\langle \mathcal{X}, \mathcal{A} \rangle)$ from (finitary) AFS $\langle \mathcal{M}, a \rangle$

1: $X_0 := L(\mathcal{M})$;
2: $Y_0 := X_0 \setminus \pi_a^+(X_0)$;
3: $G := Y_0$;
4: $k := 0$;
5: **while** $Y_k \neq \emptyset$ **do**
6: $\quad X_{k+1} := X_k \setminus (Y_k \cup \pi_a^+(Y_k))$;
7: $\quad Y_{k+1} := X_{k+1} \setminus \pi_a^+(X_{k+1})$;
8: $\quad G := G \cup Y_{k+1}$;
9: $\quad k := k + 1$;
10: **end while**
11: **report** $\mathcal{M}_G$ where $L(\mathcal{M}_G) = G$.

- adding these to the extension being accumulated (in l.3 and l.8);
- repeating this process on the AF induced by the arguments remaining after removing these and those they attack (in l.6).

The process terminates when the set of unattacked arguments is empty. Let us exemplify the application of Algorithm 1 to AF_R. Starting with the set of unattacked arguments, we get $Y_0 = 000 \cdot (000)^*$ at l.2. Then, given that $\pi_a^+(Y_0) = 00 \cdot (000)^*$ in the first iteration of the **while** loop we get $X_1 = 0 \cdot (000)^*$ at l.6 and, given that $\pi_a^+(X_1) = 00 \cdot (000)^*$, we get $Y_1 = X_1$ at l.7 and $G = (000 \cdot (000)^*) \cup (0 \cdot (000)^*)$ then in the next iteration $X_2 = Y_2 = \emptyset$ and the algorithm terminates.

Algorithm 1, however, does not guarantee termination, since it terminates only in the cases where there is k such that $\mathcal{F}^{k+1}(\emptyset) = \mathcal{F}^k(\emptyset)$. For instance in the framework AF_U of Example 3, in which for $i \geq 1$, $\mathcal{F}_i = \{\ 0^{2k-1}\ :\ 1 \leq k \leq i\}$, no such k exists.

In such cases we can use properties of the operations involved in defining attack expressions together with known identities for regular expressions to derive the form taken by arguments in the grounded extension directly. In the case of AF_U we recall that,

$$\begin{array}{lcl} \mathcal{X} & = & \{\ 0^i\ :\ i \geq 1\} \\ a & = & tl(I) \\ \pi_a^-(S) & = & tl(S) \\ \pi_a^+(S) & = & 0 \cdot S \end{array}$$

For notational ease we write $\overline{T}$ for $\mathcal{X} \setminus T$ so that $\mathcal{F}_0 = \emptyset$, $\mathcal{F}_i = \mathcal{F}(\mathcal{F}_{i-1})$ and

$$\mathcal{F}(T) \quad = \quad \overline{\pi_a^+(\ \overline{\pi_a^+(T)}\ )}$$

For AF_U we get

$$\begin{array}{lcl}
\mathcal{F}_1 & = & \overline{0 \cdot \overline{0 \cdot \emptyset}} \\
 & = & \overline{0 \cdot \mathcal{X}} \\
 & = & \{0\} \\
\mathcal{F}_2 & = & \overline{0 \cdot \overline{0 \cdot 0}} \\
 & = & \overline{\{0 \cdot 0\} \cup \{0 \cdot 0 \cdot 0 \cdot 0 \cdot 0^k \ : \ k \geq 0\}} \\
 & = & \{0, 0 \cdot 0 \cdot 0\} \\
 & \ldots & \\
\mathcal{F}_{i+1} & = & \overline{0 \cdot \overline{\{0^{2k} \ : \ 1 \leq k \leq i\}}} \\
 & = & \overline{0 \cdot \{\{0^{2k-1} \ : \ 1 \leq k \leq i\} \cup \{0^{2i+1} \cdot 0^j \ : \ j \geq 0\}\}} \\
 & = & \overline{\{0^{2k} \ : \ 1 \leq k \leq i\} \cup \{0^{2i+2} \cdot 0^j \ : \ j \geq 0\}} \\
 & = & \{0^{2k-1} \ : \ 1 \leq k \leq i+1\}
\end{array}$$

So that the least fixed-point is the infinite set $\{0^{2i-1} \ : \ i \geq 1\}$: note that, by the analysis given, this *is* a fixed-point and it is straightforward to show that no strict subset defines a fixed-point.

The analysis in the infinite case of those problems which are computationally intractable within finite AFs (cf. the summary presented in Fact 1) is left to future work. As a first step in this direction, we can show that, restricting again to finitary frameworks, limited decision procedures are possible for two of these problems. Note that we currently have no result on whether these problems are decidable: Theorem 4 ensures that, if this is not the case, they are at least semi-decidable.

Theorem 4. Let $\langle \mathcal{M}, a \rangle$ be an AFS in which the induced argumentation framework $\langle \mathcal{X}, \mathcal{A} \rangle$ is finitary. The following problems are all *semi-decidable*.

a. Determining if $\mathcal{E}_{stab}(\langle \mathcal{X}, \mathcal{A} \rangle) = \emptyset$.
b. Given a *finite* $R \subset \mathcal{X}$, determining if $\forall\ w \in R\ \neg \mathrm{CA}_{adm}(\langle \mathcal{X}, \mathcal{A} \rangle, w)$.

We note that the restriction to *finitary* frameworks (and finite subsets of $\mathcal{X}$ in the second part) is needed: without this the method used in proving Thm. 4 could not be applied.

To conclude this section, as some of the results we provided rely on the condition that an AF specification gives rise to a finitary argumentation framework, one is interested in conditions ensuring that this holds. We provide an easy sufficient condition to this purpose, namely the absence of the $*$ operator in the attack expression, leaving further investigations on this specific question for future work.

Proposition 9. Let $\langle \mathcal{M}, a \rangle$ be an AFS with induced argumentation framework $\langle \mathcal{X}, \rightarrow_a \rangle$. Let $\mathcal{K} = \{K^1_\Sigma, K^2_\Sigma, \ldots, K^r_\Sigma\}$ be the set of regular expressions used in defining a, i.e. with operations $b \cdot K_\Sigma$, $K_\Sigma \cdot b$, K_Σ / b, b / K_Σ and $b \cap K_\Sigma$. If *no* $K \in \mathcal{K}$ uses the $*$ operator then $\langle \mathcal{X}, \rightarrow_a \rangle$ is finitary.

It is easy to see that the condition of Proposition 9 is not a *necessary* one. Consider, for example $a = hd(I \cdot (\sigma_1 + \sigma_2)^*)$. This is finitary, however, fails to

satisfy the conditions of Propn. 9. Less trivially, $a = (\sigma_1 \cdot (\sigma_1)^*) \cdot I \cdot (\sigma_2 \cdot (\sigma_2)^*)$ will be finitary when $\mathcal{X} \cap \{\sigma_1 \cdot (\sigma_1)^* \cdot x \cdot \sigma_2 \cdot (\sigma_2)^*\}$ is bounded for all $x \in \mathcal{X}$.

7. AF Specifications at Work

In this section we illustrate the suitability of our approach by showing how it can be used to provide a formal representation of four examples which altogether combine different features. Two of the examples (presented in sections 7.1 and 7.2) are abstract in nature (they have been previously introduced in the literature mainly for the sake of theoretical analysis), while the examples of sections 7.3 and 7.4 are inspired to realistic application domains (also taken from the literature) and have been introduced in Section 2.2. The two abstract examples of sections 7.1 and 7.2 concern infinite non-finitary AFs, while the "application-oriented" examples of sections 7.3 and 7.4 give rise to infinite finitary AFs.

Moreover, we remark that the examples of sections 7.2 and 7.4 are based on a formalization in terms of a logic program featuring an infinite Herbrand base. In fact, the formalization in argumentation terms of logic programs with infinite Herbrand base is, at a general level, one of the "natural" applications of the proposed framework, given that a direct correspondence between logic programs with negation as failure and abstract argumentation frameworks has been established in Dung's paper itself.

We will present the abstract examples before, as they are more suitable to illustrate in detail the technical use of the formalism as a specification tool in articulated frameworks, and later the "application-oriented" examples, to give an account of some potential practical uses of the formalism without cluttering the description of the more realistic examples with too much technical details, derivable by analogy from the first examples.

7.1. The AF from Caminada and Verheij [30]

In [30] Caminada and Verheij describe a (non-finitary) AF, $\langle \mathcal{X}, \mathcal{A} \rangle$ (see Figure 3) with the property that $\langle \mathcal{X}, \mathcal{A} \rangle$ has no semi-stable extension[19], i.e. admissible set S for which $S \cup S^+$ is maximal wrt $\subseteq$. The construction uses arguments

$$\mathcal{X} \;=\; \{\, A_i \;:\; i \geq 1 \,\} \;\cup\; \{\, B_i \;:\; i \geq 1 \,\} \cup \{\, C_i \;:\; i \geq 1 \,\}$$

linked by the attack relation, $\mathcal{A}$, containing

$$\begin{array}{l} \{\, \langle A_i, A_i \rangle \;:\; i \geq 1 \,\} \;\cup\; \{\, \langle B_j, A_i \rangle \;:\; j \geq i \geq 1 \,\} \;\cup \\ \{\, \langle B_i, C_i \rangle \;:\; i \geq 1 \,\} \;\cup\; \{\, \langle B_j, B_i \rangle \;:\; j > i \geq 1 \,\} \;\cup \\ \{\, \langle C_i, B_i \rangle \;:\; i \geq 1 \,\} \end{array}$$

[19] The existence of semi-stable extensions in finitary argumentation frameworks is analyzed in [91].

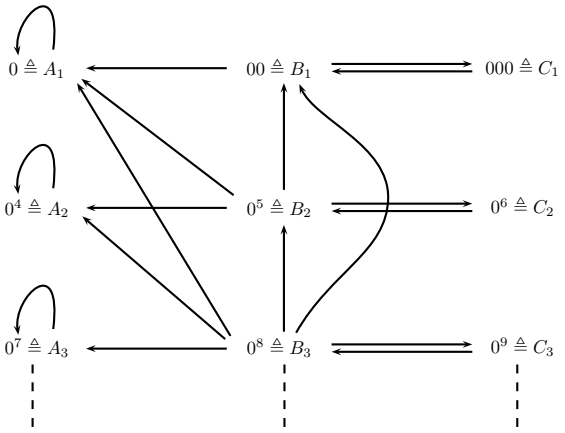

Figure 3: An infinite AF with no semi-stable extensions.

One obvious choice to describe this scheme would be the set $\Sigma = \{A, B, C\}$ so that

$$\mathcal{X} = \{A\} \cdot \{A\}^* \cup \{B\} \cdot \{B\}^* \cup \{C\} \cdot \{C\}^*$$

It is not too hard to see, however, that it is impossible to describe the required set of attacks via some $a \in \mathcal{AE}(\{A, B, C\})$: for example suppose S is an infinite regular subset of $\{A^i : i \geq 1\}$. Then

$$\pi_a^-(S) = \{ B^j : j \geq \min\{i : A^i \in S\}\} \cup S$$

Now while this is a regular language for any *fixed* subset of $\{A^i : i \geq 1\}$ given that it requires determining $\min\{ i : A^i \in S\}$ it is not possible to construct a *general* expression allowing this minimum to be computed.

Nevertheless this scheme can be described within our formalism. Let $\Sigma = \{0\}$ and $\mathcal{X} = \{0^i : i \geq 1\} = \{0\} \cdot \{0\}^*$. We can partition $\mathcal{X}$ into three sets, L_A, L_B and L_C, as follows:

$$\begin{array}{lcll} L_A & = & \{0^{3i+1} : i \geq 0\} & = \{0\} \cdot \{000\}^* \\ L_B & = & \{0^{3i+2} : i \geq 0\} & = \{00\} \cdot \{000\}^* \\ L_C & = & \{0^{3i} : i \geq 1\} & = \{000\} \cdot \{000\}^* \end{array}$$

We can now identify the attack expression:

- each element of L_A is attacked by itself, giving rise to the sub-expression $I \cap L_A$, and by all elements of L_B with greater or equal index, giving rise to the sub-expression $(I \cap L_A) \cdot 0 \cdot (000)^*$;
- each element of L_B is attacked by the element of L_C with the same index, giving rise to the sub-expression $(I \cap L_B) \cdot 0$, and by all elements of L_B with greater index, giving rise to the sub-expression $(I \cap L_B) \cdot (000) \cdot (000)^*$;

- each element of L_C is attacked by the element of L_B with the same index, giving rise to the sub-expression $tl(I \cap L_C)$.

Summing up, we get

$$\begin{array}{rcl} a & = & (I \cap L_A) \ \cup \ (I \cap L_A) \cdot 0 \cdot (000)^* \ \cup \\ & & (I \cap L_B) \cdot (000) \cdot (000)^* \ \cup \ (I \cap L_B) \cdot 0 \ \cup \\ & & tl(I \cap L_C) \end{array}$$

giving directly

$$\begin{array}{rcl} \pi_a^-(S) & = & S \cap L_A \ \cup \ (S \cap L_A) \cdot 0 \cdot (000)^* \cup \\ & & (S \cap L_B) \cdot (000) \cdot (000)^* \ \cup \ (S \cap L_B) \cdot 0 \cup \\ & & tl(S \cap L_C) \end{array}$$

In order to compute $\pi_a^+(S)$, we need to take into account Fact 2. From 2.2 the inverted mapping will be the union of the inverted sub-mappings corresponding to the various sub-expressions of a.

- The first term: $I \cap L_A$, according to 2.7 with $b = I$, gives rise to $\underline{I}^+(S \cap L_A)$, and thus, from 2.1, to $S \cap L_A$.
- As to the second term $(I \cap L_A) \cdot 0 \cdot (000)^*$, from 2.3 with $b = (I \cap L_A)$ we obtain $\underline{b}^+(S/\{0 \cdot (000)^*\})$. In turn, to obtain $\underline{b}^+$, 2.7 applies, which letting $c = I$ yields $\underline{c}^+(L_A \cap (S/\{0 \cdot (000)^*\}))$, which applying 2.1 yields $L_A \cap (S/\{0 \cdot (000)^*\})$.
- Following the same steps, from $(I \cap L_B) \cdot (000) \cdot (000)^*$ we obtain $L_B \cap (S/\{000 \cdot (000)^*\})$, and from $(I \cap L_B) \cdot 0$ we derive $L_B \cap (S/0)$.
- Finally from $tl(I \cap L_C)$, applying 2.9 with $b = I \cap L_C$ we obtain $\underline{b}^+(\Sigma \cdot S)$. Applying then 2.7 with $c = I$ we get $\underline{c}^+(L_C \cap \Sigma \cdot S)$, which applying 2.1 yields $L_C \cap \Sigma \cdot S$.

Putting things together (according to Fact 2.2), we obtain

$$\begin{array}{rcl} \pi_a^+(S) & = & (S \cap L_A) \ \cup \ (L_A \cap (S/\{0 \cdot (000)^*\})) \cup \\ & & (L_B \cap (S/\{000 \cdot (000)^*\})) \cup (L_B \cap (S/0)) \cup \\ & & (L_C \cap (\Sigma \cdot S)) \end{array}$$

We can now exemplify the use of the computational procedures[20] of Section 6 in this case. Let us start with the check of conflict-freeness, which for a set S involves verifying whether $\pi_a^-(S) \cap S \ = \ \emptyset$. From the formulation of $\pi_a^+(S)$ given above it is easily verifiable that:

- any set S such that $S \cap L_A \neq \emptyset$ is not conflict-free;
- any set S such that $|S \cap L_B| \geq 2$ is not conflict-free;

[20] The reader is referred to the proof of Theorem 3 for the underlying details.

- any set $S \subseteq L_C$ is conflict-free.

As to conflict-freeness, leaving apart the empty set, we have therefore to consider only singletons of the form $S = \{00 \cdot (000)^i\}$ with a fixed $i \geq 0$, any set $S \subseteq L_C$, and the union of any of the singletons $\{00 \cdot (000)^i\}$ with any subset of $L_C \setminus \{000 \cdot (000)^i\}$. For admissibility of a set S, one has to verify whether $\pi_a^+(S) \supseteq \pi_a^-(S)$ in addition to conflict-freeness. For the generic singleton $S = \{00 \cdot (000)^i\}$, we have $\pi_a^+(S) = \{0 \cdot (000)^j\} \cup \{00 \cdot (000)^k\} \cup \{000 \cdot (000)^i\}$ for $0 \leq j \leq i$ and $0 \leq k < i$, while $\pi_a^-(S) = \{00 \cdot (000)^{i+1} \cdot (000)^*\} \cup \{000 \cdot (000)^i\}$. Since $\{00 \cdot (000)^{i+1} \cdot (000)^*\} \cap \pi_a^+(S) = \emptyset$, it turns out that $\pi_a^+(S) \not\supseteq \pi_a^-(S)$ hence no $S = \{00 \cdot (000)^i\}$ is admissible. Considering instead any set $S \subseteq L_C$ it can be seen that $\pi_a^+(S) = L_B \cap (S/0) = tl(S \cap L_C) = \pi_a^-(S)$. Hence any subset of L_C is admissible. Considering now the union S of a singleton $\{00 \cdot (000)^i\}$ and a subset of L_C, since $\pi_a^-(\{00 \cdot (000)^i\}) \supseteq \{00 \cdot (000)^{i+1} \cdot (000)^*\}$ then $\{(000)^{i+2} \cdot (000)^*\}$ must be contained in S. Taking now into account the facts above, it turns out that $S = \{00 \cdot (000)^i, (000)^{i+2} \cdot (000)^*\} \cup L'$ for any $L' \subseteq \{(000)^j | 1 \leq j \leq i\}$. To determine whether an admissible set $S \subseteq L_C$ is a complete extension we have to check whether $S = \mathcal{F}(S) = \mathcal{X} \setminus \pi_a^+(\mathcal{X} \setminus \pi_a^+(S))$. Considering any such $S \subseteq L_C$, we have already seen that $\pi_a^+(S) = L_B \cap (S/0)$. Hence $\mathcal{X} \setminus \pi_a^+(S) = L_A \cup (L_B \setminus (S/0)) \cup L_C$. It follows that $\pi_a^+(\mathcal{X} \setminus \pi_a^+(S)) = L_A \cup L_B \cup (L_C \setminus S)$, and hence $\mathcal{F}(S) = \mathcal{X} \setminus (L_A \cup L_B \cup (L_C \setminus S)) = S$. As to the sets of the form $S = \{00 \cdot (000)^i, (000)^{i+2} \cdot (000)^*\} \cup L'$, it turns out (using the same reasoning line) that $\mathcal{F}(S) = \{00 \cdot (000)^i\} \cup (L_C \setminus \{000 \cdot (000)^i\})$, which is also the unique complete extension including $\{00 \cdot (000)^i\}$. Summing up, all (either finite or infinite) subsets of L_C plus the sets $\{00 \cdot (000)^i\} \cup (L_C \setminus \{000 \cdot (000)^i\})$ with $i \geq 0$ form the set of all the complete extensions of this framework.

7.2. Dung's example

We will now consider the infinite argumentation framework introduced in [37, p. 331, 352]. The framework is derived from the following logic program[21] Q.

$$
\begin{array}{lll}
Q: & r \leftarrow not\ p & (r1) \\
 & p \leftarrow not\ q(x) & (r2) \\
 & q(x) \leftarrow even(x) & (r3) \\
 & q(x) \leftarrow not\ even(x) & (r4) \\
 & even(s(x)) \leftarrow not\ even(x) & (r5) \\
 & even(0) \leftarrow & (r6)
\end{array}
$$

The rules for transforming a logic program into an AF are defined in [37, p. 343] as follows.

First of all, for a logic program P, G_P denotes the *set of all ground instances of clauses* in P. For each literal h, the *complement* of h is denoted by h^*. Let $K = \{not\ b_1, \ldots, not\ b_m\}$ be a set of ground negative literals. A ground atom

[21] In the logic program Q, x is any natural number, $s(x)$ denotes the successor of x and $q(x)$ can be regarded as any property of all natural numbers.

k is said to be a *defeasible consequence* of P, K if there is a sequence of ground atoms $(e_0, e_1, \ldots, e_n)$ with $e_n = k$ such that for each e_i, either $e_i \leftarrow \ \in G_P$ or e_i is the head of a clause $e_i \leftarrow a_1, \ldots, a_t, not\ a_{t+1}, \ldots, not\ a_{t+r}$ in G_P such that the positive literals $a_1, \ldots, a_t$ belong to the preceding members in the sequence and the negative literals $not\ a_{t+1}, \ldots, not\ a_{t+3}$ belong to K. K is said to be a *support for k with respect to P*.

A logic program P is transformed into an AF $AF(P) = \langle \mathcal{X}_P, \mathcal{A}_P \rangle$ as follows:

$$\begin{array}{rcl} \mathcal{X}_P & = & \{(K,k) | K \text{ is a support for } k \text{ with respect to } P\} \\ & & \cup \{(\{not\ k\}, not\ k) | k \text{ is a ground atom}\}; \\ (K,h) \text{ attacks } (K',h') & \Leftrightarrow & h^* \in K' \end{array}$$

Arguments of the form $(\{not\ k\}, not\ k)$ capture the idea that k would be concluded false if there is no acceptable argument supporting k. An argument a attacks an argument b when the consequence of a contradicts one of the members of the support of b.

The framework $AF(Q) = \langle \mathcal{X}_Q, \mathcal{A}_Q \rangle$, derived from the logic program Q, turns out to be non finitary and is depicted in Figure 4. Note that to keep the notation simple each predicate $s(\bar{x})$ has been replaced by the result of the expression $\bar{x}+1$.

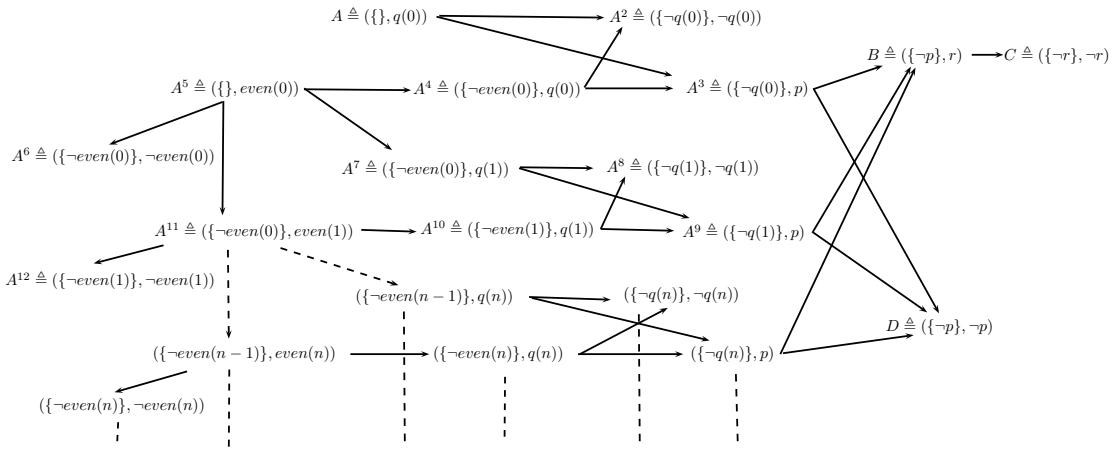

Figure 4: Graphical representation of $AF(Q)$.

Let us examine the elements of $\mathcal{X}_Q$. First, there are arguments of the form $(\{not\ k\}, not\ k)$ for each ground atom k, namely:

- two "single" arguments for the atoms p and r (at the right of the figure, respectively in the lower and higher part)
- an infinite sequence of arguments for the atoms with form $q(\bar{x})$, corresponding to the fifth (from the left) "column" in Figure 4.
- an infinite sequence of arguments for the atoms with form $even(\bar{x})$, corresponding to the first (from the left) "column" in Figure 4

Then, there are arguments of the form (K, k) derived by applying rules (r1)-(r6), where K is a support for k. In particular we have:

- two arguments with empty support: from (r6) we get $(\{\}, even(0))$ (top of the second "column") and from (r6) and (r3) we get $(\{\}, q(0))$ (above third "column" in Figure 4)
- a "single" argument $(\{not\ p\}, r)$ from (r1) (at the right of the figure, in the higher part)
- an infinite sequence of arguments of the form $(\{not\ q(\bar{x})\}, p)|\bar{x} \geq 0$ from (r2) (sixth "column")
- an infinite sequence of arguments of the form $(\{not\ even(\bar{x})\}, q(\bar{x}))|\bar{x} \geq 0$ from (r4) (fourth "column")
- an infinite sequence of arguments of the form $(\{not\ even(\bar{x})\}, even(\bar{x} + 1))|\bar{x} \geq 0$ from (r5) (second "column")
- an infinite sequence of arguments of the form $(\{not\ even(\bar{x})\}, q(\bar{x}+1))|\bar{x} \geq 0$ from (r5) and (r3) (third "column")

Turning to the attack relation $\mathcal{A}_Q$, we observe that:

- each argument in the second "column", (with consequence $even(\bar{x})$) attacks the corresponding arguments in the first, third and fourth "columns" (having support $not\ even(\bar{x})$)
- each argument in the second "column" also attacks its "successor" in the column
- each argument in the third "column", (with consequence $q(\bar{x})$) attacks the corresponding arguments in the fifth and sixth "columns" (having support $not\ q(\bar{x})$)
- similarly, each argument in the fourth "column", (with consequence $q(\bar{x})$) attacks the corresponding arguments in the fifth and sixth "columns" (having support $not\ q(\bar{x})$)
- each argument in the sixth "column" (with consequence p) attacks both arguments $(\{not\ p\}, not\ p)$ and $(\{not\ p\}, r)$
- argument $(\{not\ p\}, r)$ attacks $(\{not\ r\}, not\ r)$

In order to provide an AF specification for $AF(Q)$ we need first a DFA representing the infinite set of arguments $\mathcal{X}_Q$ and then a proper attack expression representing the relation $\mathcal{A}_Q$.

As to the DFA representation of $\mathcal{X}_Q$, it is handy to consider separately the infinite sequences corresponding to the six "columns" in Figure 4 and the three "single" arguments $(\{not\ p\}, r)$, $(\{not\ r\}, not\ r)$, and $(\{not\ p\}, not\ p)$.

As to the arguments included in the "columns", we define a correspondence with sequences of a unique symbol, namely A, by exploiting the "regular" structure of the sequences of arguments. In fact, let us associate the string A with argument $(\{\}, q(0))$, AA with argument $(\{not\ q(0)\}, not\ q(0))$, AAA with argument $(\{not\ q(0)\}, p)$, $AAAA$ with argument $(\{not\ even(0)\}, q(0))$, A^5 with argument $(\{\}, even(0))$, and A^6 with argument $(\{not\ even(0)\}, not\ even(0))$. The association may then continue periodically over the "columns" by putting A^7 in correspondence with $(\{not\ even(0)\}, q(1))$, A^8 with $(\{not\ q(1)\}, not\ q(1))$ and so on. More formally, we use the elements of $A \cdot A^*$ to represent arguments as follows:

$$
\begin{array}{l}
A \triangleq (\{\}, q(0)) \\
\forall \bar{x} \geq 0, A^{2+6\bar{x}} \triangleq (\{not\ q(\bar{x})\}, not\ q(\bar{x})) \\
\forall \bar{x} \geq 0, A^{3+6\bar{x}} \triangleq (\{not\ q(\bar{x})\}, p) \\
\forall \bar{x} \geq 0, A^{4+6\bar{x}} \triangleq (\{not\ even(\bar{x})\}, q(\bar{x})) \\
A^5 \triangleq (\{\}, even(0)) \\
\forall \bar{x} \geq 0, A^{6+6\bar{x}} \triangleq (\{not\ even(\bar{x})\}, not\ even(\bar{x})) \\
\forall \bar{x} \geq 0, A^{7+6\bar{x}} \triangleq (\{not\ even(\bar{x})\}, q(\bar{x}+1)) \\
\forall \bar{x} \geq 0, A^{11+6\bar{x}} \triangleq (\{not\ even(\bar{x})\}, even(\bar{x}+1))
\end{array}
$$

To simplify the subsequent description it is useful to denote as six distinct (regular) languages the subsets of A^* corresponding to the different sequences of arguments in $\mathcal{X}_Q$:

$$
\begin{array}{l}
L_2 \triangleq \{A^{2+6\bar{x}} | \bar{x} \geq 0\} \\
L_3 \triangleq \{A^{3+6\bar{x}} | \bar{x} \geq 0\} \\
L_4 \triangleq \{A^{4+6\bar{x}} | \bar{x} \geq 0\} \\
L_6 \triangleq \{A^{6+6\bar{x}} | \bar{x} \geq 0\} \\
L_7 \triangleq \{A^{7+6\bar{x}} | \bar{x} \geq 0\} \\
L_{11} \triangleq \{A^{11+6\bar{x}} | \bar{x} \geq 0\}
\end{array}
$$

To complete the representation of $\mathcal{X}_Q$, the remaining three "single" arguments are assigned three distinct alphabet elements as follows:

- $B \triangleq (\{not\ p\}, r)$
- $C \triangleq (\{not\ r\}, not\ r)$
- $D \triangleq (\{not\ p\}, not\ p)$

In summary, the representation of the arguments in $AF(Q)$ is based on the set of symbols $\Sigma_Q = \{A, B, C, D\}$ and the encoding consists of a DFA $\mathcal{M}_Q$ accepting the language $L(\mathcal{M}_Q) = \{B, C, D\} \cup \{A \cdot A^*\}$. Clearly $L(\mathcal{M}_Q)$ is a regular language included in $\Sigma_Q^* \setminus \{\varepsilon\}$.

The simple minimal DFA accepting $L(\mathcal{M}_Q)$ is depicted in Figure 5.

We need now to identify a suitable attack expression a^Q for $AF(Q)$. To this purpose let us first observe that, for a set of arguments S, the function $\pi^-_{a^Q}(S)$ must return as result a non-empty set if and only if S has a non empty intersection with the set of arguments which receive an attack in $AF(Q)$, namely $\{B, C, D\} \cup L_2 \cup L_3 \cup L_4 \cup L_6 \cup L_7 \cup L_{11}$. As a consequence, the global attack

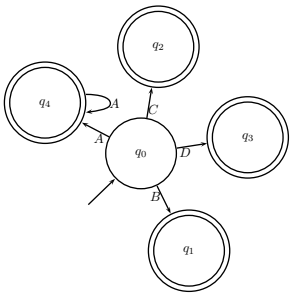

Figure 5: Graphical representation of the minimal DFA describing the argument encoding of $AF(Q)$.

expression may be built as the union of various sub-expressions, each associated with a class of attacked arguments. Each sub-expression:

- has to select the range of sets for which a non-empty result is returned: this can be achieved specifying the intersection between I and a given sub-language of $\mathcal{X}_Q$ corresponding to the class of attacked arguments;
- has to define a set of attackers through a proper symbol manipulation.

To exemplify, consider a sub-expression to specify the attackers of argument B, namely $\{A^{3+6\bar{x}} | \bar{x} \geq 0\}$ (corresponding to $\{(\{not\ q(\bar{x})\}, p)\}$, $\bar{x} \geq 0$). This can be obtained as $tl(I \cap B) \cdot AAA \cdot (\underbrace{A \cdot \ldots \cdot A}_{6})^* = tl(I \cap B) \cdot L_3$. In fact, $I \cap B$ yields either $\{B\}$ or the empty set. In the former case the tl operator yields ε which, combined with L_3, gives the desired result, while in the latter case the combination produces the empty set. In a similar (and simpler) way, since C is attacked only by B we obtain the subexpression $tl(I \cap C) \cdot B$, while D has exactly the same attackers as B, yielding $tl(I \cap D) \cdot L_3$.

Consider now the specification of the attackers of the elements of sub-languages of $\mathcal{X}_Q$. Starting from L_2 we observe that each element $A^{2+6\bar{x}}$ (corresponding to $(\{not\ q(\bar{x})\}, not\ q(\bar{x}))$, $\bar{x} \geq 0$) has two attackers namely $A^{2+6\bar{x}-1}$ (corresponding to $(\{\}, q(0))$ for $\bar{x} = 0$ and to $(\{not\ even(\bar{x}-1)\}, q(\bar{x}))$ for $\bar{x} > 0$) and $A^{2+6\bar{x}+2}$ (corresponding to $(\{not\ even(\bar{x})\}, q(\bar{x}))$, $\bar{x} \geq 0$). The elements of the first family of attackers can be obtained by applying the tl operator to L_2, those of the second family of attackers by concatenating $A \cdot A$ to L_2. This reasoning gives rise to the sub-expressions $tl(I \cap L_2)$ and $(I \cap L_2) \cdot A \cdot A$.

Similarly, each element of L_3 $A^{3+6\bar{x}}$ (corresponding to $(\{not\ q(\bar{x})\}, p)$, $\bar{x} \geq 0$) has two attackers namely $A^{3+6\bar{x}+1}$ (corresponding to $(\{not\ even(\bar{x})\}, q(\bar{x}))$, $\bar{x} \geq 0$) and $A^{3+6\bar{x}-2}$ (corresponding to $(\{\}, q(0))$ for $\bar{x} = 0$ and to $(\{not\ even(\bar{x}-1)\}, q(\bar{x}))$ for $\bar{x} \geq 1$). This reasoning gives rise to the sub-expressions[22] $(I \cap L_3) \cdot A$ and $tl^2(I \cap L_3)$.

[22] In the following, in order to simplify notation, we denote as $tl^n(w)$ the n^{th} application of

As to L_4, each element $A^{4+6\bar{x}}$ (corresponding to $(\{not\ even(\bar{x})\}, q(\bar{x}))$, $\bar{x} \geq 0$) has one attacker $A^{4+6\bar{x}+1}$ (corresponding to $(\{\}, even(0))$ for $\bar{x} = 0$ and to $(\{not\ even(\bar{x}-1)\}, even(\bar{x}))$, $\bar{x} \geq 1$). This gives rise to the sub-expression $(I \cap L_4) \cdot A$.

Turning to L_6, each element $A^{6+6\bar{x}}$ (corresponding to $(\{not\ even(\bar{x})\}, not\ even(\bar{x}))$, $\bar{x} \geq 0$) has one attacker $A^{6+6\bar{x}-1}$ (corresponding to $(\{\}, even(0))$ for $\bar{x} = 0$ and to $(\{not\ even(\bar{x}-1)\}, even(\bar{x}))$, $\bar{x} \geq 1$). This gives rise to the sub-expression $tl(I \cap L_6)$.

In L_7, each element $A^{7+6\bar{x}}$ (corresponding to $(\{not\ even(\bar{x})\}, q(\bar{x}+1))$, $\bar{x} \geq 0$) has one attacker $A^{7+6\bar{x}-2}$ (corresponding to $(\{\}, even(0))$ for $\bar{x} = 0$ and to $(\{not\ even(\bar{x}-1)\}, even(\bar{x}))$, $\bar{x} \geq 1$). This gives rise to the sub-expression $tl^2(I \cap L_7)$.

Finally, each element of L_{11} $A^{11+6\bar{x}}$,(corresponding to $(\{not\ even(\bar{x})\}, even(\bar{x}+1))$, $\bar{x} \geq 0$) has one attacker $A^{11+6\bar{x}-6}$ (corresponding to $(\{\}, even(0))$ for $\bar{x} = 0$ and to $(\{not\ even(\bar{x}-1)\}, even(\bar{x}))$, $\bar{x} \geq 1$). This gives rise to the sub-expression $tl^6(I \cap L_{11})$.

In summary, we obtain the following attack expression:

$$\begin{aligned} a^Q \quad \triangleq \quad & tl(I \cap B) \cdot L_3 \ \cup \\ & tl(I \cap C) \cdot B \ \cup \\ & tl(I \cap D) \cdot L_3 \ \cup \\ & tl(I \cap L_2) \ \cup \\ & (I \cap L_2) \cdot A \cdot A \ \cup \\ & tl^2(I \cap L_3) \ \cup \\ & (I \cap L_3) \cdot A \ \cup \\ & (I \cap L_4) \cdot A \ \cup \\ & tl(I \cap L_6) \ \cup \\ & tl^2(I \cap L_7) \ \cup \\ & tl^6(I \cap L_{11}) \end{aligned}$$

The relevant mapping $\underline{a^Q} : 2^{\Sigma_Q^*} \to 2^{\Sigma_Q^*}$ follows directly:

$$\begin{aligned} \underline{a^Q}(S) \quad = \quad & tl(S \cap B) \cdot L_3 \ \cup \\ & tl(S \cap C) \cdot B \ \cup \\ & tl(S \cap D) \cdot L_3 \ \cup \\ & tl(S \cap L_2) \ \cup \\ & (S \cap L_2) \cdot A \cdot A \ \cup \\ & tl^2(S \cap L_3) \ \cup \end{aligned}$$

$tl(\cdot)$ to the word w, namely $tl(tl(\ldots(tl(w))))$.

$$
\begin{array}{l}
(S \cap L_3) \cdot A \ \cup \\
(S \cap L_4) \cdot A \ \cup \\
tl(S \cap L_6) \ \cup \\
tl^2(S \cap L_7) \ \cup \\
tl^6(S \cap L_{11})
\end{array}
$$

We can now apply Fact 2 to obtain the inverted mapping $\underline{a^Q}^+$. First, we observe that on the basis of 2.2 the inverted mapping will be the union of the inverted sub-mappings corresponding to the various sub-expressions of a^Q.

Consider first, for the sake of illustration, the sub-expression $tl(I \cap B) \cdot L_3$, which has the form $b \cdot K_\Sigma$ with $b = tl(I \cap B)$ and $K_\Sigma = L_3$. Accordingly, Fact 2.3 applies, yielding $\underline{b}^+(S/L_3)$. In turn, to obtain $\underline{b}^+$, Fact 2.9 applies which, letting $c = I \cap B$, gives $\underline{c}^+(\Sigma_Q \cdot (S/L_3))$. Applying Fact 2.7 (and the base case for I) to c we obtain $B \cap (\Sigma_Q \cdot (S/L_3))$.

The sub-expressions $tl(I \cap C) \cdot B$ and $tl(I \cap D) \cdot L_3$ are analogous, yielding $C \cap (\Sigma_Q \cdot (S/B))$ and $D \cap (\Sigma_Q \cdot (S/L_3))$.

From the sub-expression $tl(I \cap L_2)$ orderly applying Fact 2.9 and 2.7 we obtain $L_2 \cap (\Sigma_Q \cdot S)$, while from $(I \cap L_2) \cdot A \cdot A$ applying 2.3 and 2.7 we have $L_2 \cap (S/(A \cdot A))$.

For the sub-expression $tl^2(I \cap L_3)$ we apply 2.9 twice and 2.7, yielding $L_3 \cap (\Sigma_Q \cdot \Sigma_Q \cdot S)$.

The treatment of each of the remaining sub-expressions is similar to one of the previous cases, yielding the following result.

$$
\begin{array}{rcl}
\underline{a^Q}^+(S) & = & B \cap (\Sigma_Q \cdot (S/L_3)) \ \cup \\
& & C \cap (\Sigma_Q \cdot (S/B)) \ \cup \\
& & D \cap (\Sigma_Q \cdot (S/L_3)) \ \cup \\
& & L_2 \cap (\Sigma_Q \cdot S) \ \cup \\
& & L_2 \cap (S/(A \cdot A)) \ \cup \\
& & L_3 \cap (\Sigma_Q \cdot \Sigma_Q \cdot S) \ \cup \\
& & L_3 \cap (S/A) \ \cup \\
& & L_4 \cap (S/A) \ \cup \\
& & L_6 \cap (\Sigma_Q \cdot S) \ \cup \\
& & L_7 \cap (\Sigma_Q \cdot \Sigma_Q \cdot S) \ \cup \\
& & L_{11} \cap (\Sigma_Q \cdot \Sigma_Q \cdot \Sigma_Q \cdot \Sigma_Q \cdot \Sigma_Q \cdot \Sigma_Q \cdot S)
\end{array}
$$

It can be easily observed that both $\underline{a^Q}$ and $\underline{a^Q}^+$ can not produce results outside $\mathcal{X}_Q$ hence, for any $S \subseteq \mathcal{X}_Q$, $\pi^-_{a^Q}(S) = \underline{a^Q}(S) \cap \mathcal{X}_Q = \underline{a^Q}(S)$ and $\pi^+_{a^Q}(S) = \underline{a^Q}^+(S) \cap \mathcal{X}_Q = \underline{a^Q}^+(S)$.

We can now exemplify the analysis of semantics properties in $AF(Q)$.

Letting $S = \{B\} \cup \{D\} \cup \{A^{1+12\bar{x}} | \bar{x} \geq 0\} \cup \{A^{5+12\bar{x}} | \bar{x} \geq 0\} \cup \{A^{10+12\bar{x}} | \bar{x} \geq 0\} \cup \{A^{12+12\bar{x}} | \bar{x} \geq 0\}$ consider the problem of checking whether S is conflict-free (the structure of the infinite set S is evidenced in Figure 6).

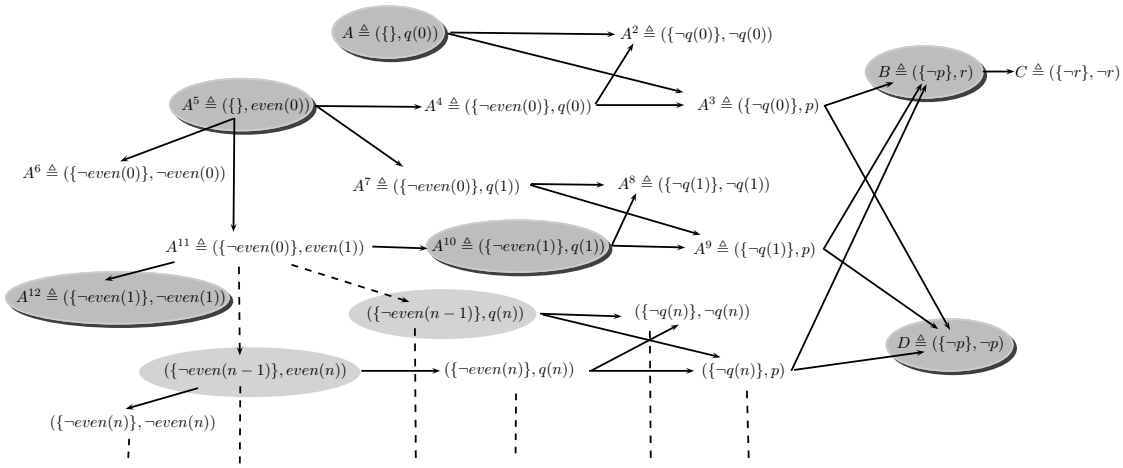

Figure 6: $AF(Q)$ with an infinite subset evidenced.

We have to prove that $\pi^{-}_{a^Q}(S) \cap S = \emptyset$.

We can now apply $\underline{a^Q}$ to the subsets forming the definition of S. Noting in particular that

- $\{A^{1+12\bar{x}} | \bar{x} \geq 0\}$ intersects L_7 "starting" from $\underbrace{A \cdot \ldots \cdot A}_{13}$
- $\{A^{5+12\bar{x}} | \bar{x} \geq 0\}$ intersects L_{11} "starting" from $\underbrace{A \cdot \ldots \cdot A}_{17}$
- $\{A^{10+12\bar{x}} | \bar{x} \geq 0\}$ intersects L_4 "starting" from $\underbrace{A \cdot \ldots \cdot A}_{10}$
- $\{A^{12+12\bar{x}} | \bar{x} \geq 0\}$ intersects L_6 "starting" from $\underbrace{A \cdot \ldots \cdot A}_{12}$

we obtain:

$$\begin{aligned} \pi^{-}_{a^Q}(S) \quad = \quad & L_3 \cup L_3 \cup \\ & tl^2(\underbrace{A \cdot \ldots \cdot A}_{13} \cdot (\underbrace{A \cdot \ldots \cdot A}_{12})^*) \cup \\ & tl^6(\underbrace{A \cdot \ldots \cdot A}_{17} \cdot (\underbrace{A \cdot \ldots \cdot A}_{12})^*) \cup \\ & (\underbrace{A \cdot \ldots \cdot A}_{10} \cdot (\underbrace{A \cdot \ldots \cdot A}_{12})^*) \cdot A \cup \end{aligned}$$

$$tl(\underbrace{A\cdot\ldots\cdot A}_{12}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*).$$

Noting that the last four elements of the above expression coincide, we have $\pi^-_{a^Q}(S) = L_3 \cup \underbrace{A\cdot\ldots\cdot A}_{11}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*$. As to conflict-freeness, it is easily seen that $\pi^-_{a^Q}(S)\cap S = \emptyset$.

Let us now turn to the problem of acceptability checking, by verifying whether the argument B is acceptable wrt S, i.e. $B \in \mathcal{F}_Q(S)$. This requires to check whether $\pi^-_{a^Q}(\{B\}) \setminus \pi^+_{a^Q}(S) = \emptyset$ (see the proof of part b of Theorem 3).

To identify $\pi^+_{a^Q}(S)$ we can apply $\underline{a^Q}^+$ to the subsets evidenced in the above definition of S. In particular:

- the second item in the definition of $\underline{a^Q}^+$ is effective (i.e. gives a non-empty result) on $\{B\}$ yielding $\{C\}$
- no item is effective on $\{D\}$
- the fourth and sixth items are effective on $\{A^{1+12\bar{x}}|\bar{x}\geq 0\}$ yielding $A\cdot A\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*$ and $A\cdot A\cdot A\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*$
- the eighth, ninth, tenth, and eleventh items are effective on $\{A^{5+12\bar{x}}|\bar{x}\geq 0\}$ yielding $\underbrace{A\cdot\ldots\cdot A}_{4}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*$, $\underbrace{A\cdot\ldots\cdot A}_{6}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*$, $\underbrace{A\cdot\ldots\cdot A}_{7}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*$, $\underbrace{A\cdot\ldots\cdot A}_{11}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*$
- the fifth and seventh items are effective on $\{A^{10+12\bar{x}}|\bar{x}\geq 0\}$ yielding $\underbrace{A\cdot\ldots\cdot A}_{8}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*$ and $\underbrace{A\cdot\ldots\cdot A}_{9}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*$
- no item is effective on $\{A^{12+12\bar{x}}|\bar{x}\geq 0\}$

Summing up,

$$\begin{aligned}\pi^+_{a^Q}(S) \;=\; & \{C\}\cup\\ & A\cdot A\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*\cup\\ & A\cdot A\cdot A\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*\cup\\ & \underbrace{A\cdot\ldots\cdot A}_{4}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*\cup\\ & \underbrace{A\cdot\ldots\cdot A}_{6}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*\cup\\ & \underbrace{A\cdot\ldots\cdot A}_{7}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*\cup\end{aligned}$$

$$\underbrace{A\cdot\ldots\cdot A}_{11}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*\cup$$
$$\underbrace{A\cdot\ldots\cdot A}_{8}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*\cup$$
$$\underbrace{A\cdot\ldots\cdot A}_{9}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*.$$

Since $\pi^-_{a^Q}(\{B\}) = L_3$ while from the expression derived above we note that $A\cdot A\cdot A\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*\cup\underbrace{A\cdot\ldots\cdot A}_{9}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^* = L_3 \subset \pi^+_{a^Q}(S)$, it follows that $B \in \mathcal{F}_Q(S)$.

Let us now check whether S is admissible. We have already proved that S is conflict free, therefore, from part (c) of Theorem 3, we have to check whether $\pi^-_{a^Q}(S) \setminus \pi^+_{a^Q}(S) = \emptyset$.

Recalling

$$\pi^-_{a^Q}(S) \;=\; L_3 \;\cup\; \underbrace{A\cdot\ldots\cdot A}_{11}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^*$$

it is easily seen that $\pi^-_{a^Q}(S) \setminus \pi^+_{a^Q}(S) = \emptyset$.

We can also check whether S is a stable extension. Since S is conflict free, we just need to confirm that $S \cup \pi^+_a(S) = \mathcal{X}$:

$$\begin{aligned}
S\cup\pi^+_{a^Q}(S) \;=\;& B \;\cup\; D \;\cup\; A\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^* \;\cup\\
& \underbrace{A\cdot\ldots\cdot A}_{5}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^* \;\cup\; \underbrace{A\cdot\ldots\cdot A}_{10}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^* \;\cup\\
& \underbrace{A\cdot\ldots\cdot A}_{12}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^* \;\cup\\
& \{C\}\cup A\cdot A\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^* \;\cup\; A\cdot A\cdot A\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^* \;\cup\\
& \underbrace{A\cdot\ldots\cdot A}_{4}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^* \;\cup\; \underbrace{A\cdot\ldots\cdot A}_{6}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^* \;\cup\\
& \underbrace{A\cdot\ldots\cdot A}_{7}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^* \;\cup\\
& \underbrace{A\cdot\ldots\cdot A}_{8}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^* \;\cup\; \underbrace{A\cdot\ldots\cdot A}_{9}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^* \;\cup\\
& \underbrace{A\cdot\ldots\cdot A}_{11}\cdot(\underbrace{A\cdot\ldots\cdot A}_{12})^* \;\cup\\
=\;& A\cdot A^* \;\cup\; B \;\cup\; C \;\cup\; D\\
=\;& \mathcal{X}_Q
\end{aligned}$$

Then S is a stable extension of $AF(Q)$. From this fact it follows that S is also a complete extension of $AF(Q)$, hence $\mathcal{F}_Q(S) = S$. This could be independently

verified, according to part (f) of Theorem 3, computing $\mathcal{F}_Q(S) = \mathcal{X}_Q \setminus \pi^+_{a_Q}(\mathcal{X}_Q \setminus \pi^+_{a_Q}(S))$. As we already know, $\mathcal{X}_Q \setminus \pi^+_{a_Q}(S) = S$, hence $\mathcal{F}_Q(S) = \mathcal{X}_Q \setminus \pi^+_{a_Q}(S) = S$.

It can also be observed that $AF(Q)$ is well-founded (Definition 29 of [37]) namely there is no infinite sequence of arguments $X_0, X_1, \ldots, X_n, \ldots$ such that X_{i+1} attacks X_i. Note in particular that letting X_0 any argument in L_{11}, i.e. $X_0 = A^{11+6\bar{x}}$ for some $\bar{x} \geq 0$, there is only a finite sequence $X_0, \ldots, X_{\bar{x}+1}$ satisfying the condition of Definition 29 in [37], with $X_1 = A^{11+6(\bar{x}-1)}, \ldots, X_{\bar{x}+1} = A^5$. Note also that the framework would not be well-founded with a "reverse" attack relation, namely if we had $(A^{11+6(\bar{x}+1)}, A^{11+6(\bar{x})}) \in \mathcal{A}_Q$ instead of having $(A^{11+6(\bar{x}-1)}, A^{11+6(\bar{x})}) \in \mathcal{A}_Q$.

Since $AF(Q)$ is well-founded, by Theorem 30 of [37] it has exactly one complete extension which is grounded, preferred and stable, namely the set S identified above. It is described by the regular language $L_{\mathcal{E}^Q_{gr}}$ accepted by the DFA depicted in Figure 7.

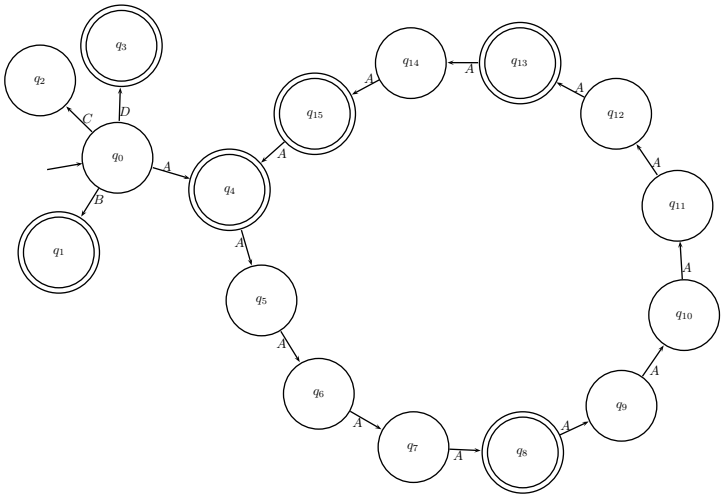

Figure 7: A DFA accepting the regular language representing the unique complete, grounded, preferred, and stable extension of $AF(Q)$.

7.3. An example in multi-agent negotiation (from Sec. 2.2.1)

Referring to the description of the example given in Section 2.2.1, the global argumentation framework AF_{neg} arising from the non-terminating message exchanges among the three agents is depicted in Figure 8.

Upon detection of a long sequence of withdrawals and reiterations of the same offers (and assuming that the agents programmatically repeat their behavior),

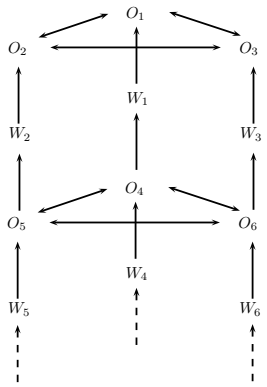

Figure 8: Graphical representation of AF_{neg}.

the market authority can identify[23] the relevant AF specification, which can be given as follows.

Let $\Sigma = \{0\}$ and $\mathcal{X} = \{0^i : i \geq 1\} = \{0\} \cdot \{0\}^*$. We can partition $\mathcal{X}$ into six sets L_A, L_B, L_C, L_D, L_E, and L_F (corresponding respectively to the six sequences $\{O_1, O_4, \ldots\}$, $\{O_2, O_5, \ldots\}$, $\{O_3, O_6, \ldots\}$, $\{W_1, W_4, \ldots\}$, $\{W_2, W_5, \ldots\}$, $\{W_3, W_6, \ldots\}$) as follows:

$$\begin{array}{lcll} L_A & = & \{0^{6i+1} : i \geq 0\} & = & \{0\} \cdot \{000000\}^* \\ L_B & = & \{0^{6i+2} : i \geq 0\} & = & \{00\} \cdot \{000000\}^* \\ L_C & = & \{0^{6i+3} : i \geq 0\} & = & \{000\} \cdot \{000000\}^* \\ L_D & = & \{0^{6i+4} : i \geq 0\} & = & \{0000\} \cdot \{000000\}^* \\ L_E & = & \{0^{6i+5} : i \geq 0\} & = & \{00000\} \cdot \{000000\}^* \\ L_F & = & \{0^{6i} : i \geq 1\} & = & \{000000\} \cdot \{000000\}^* \end{array}$$

The attack expression can then be formulated as follows:

$a = (I \cdot 000) \cup (I \cap L_A) \cdot 0 \cup (I \cap L_A) \cdot (00) \cup (I \cap L_B) \cdot 0 \cup tl((I \cap L_B)) \cup tl(I \cap L_C) \cup tl(tl(I \cap L_C))$.

The market authority can then stop the activities of the agents and check whether some combination of offers and withdrawals can be regarded as a feasible solution (the market authority is interested in favouring the execution of as many exchanges as possible). Using the algorithms presented in Section 6, it can be checked that:

- all three sets representing the reiteration of a specific offer, namely L_A, L_B, and L_C corresponding respectively to $\{O_1, O_4, O_7, \ldots\}$, $\{O_2, O_5, O_8, \ldots\}$,

[23] The problem of identifying an AF specification from a regular sequence of observations has direct connections with the the widely studied (and partially overlapping) fields of automata identification and grammatical inference [34]. Defining algorithms for the identification of AF specification is an interesting issue for future work, that we are confident can be faced resorting to techniques borrowed from the above mentioned areas.

and $\{O_3, O_6, O_9, \ldots\}$, are admissible

- none of the possible pairwise unions of the three sets above is admissible
- each set consisting of the reiteration of an offer and of the withdrawals of the two other offers (i.e. each of the following sets $L_A \cup L_E \cup L_F$; $L_B \cup L_D \cup L_F$; $L_C \cup L_D \cup L_E$;) is stable.

On the basis of these evaluations, it emerges that exactly one of the three exchanges can be executed, with the choice left to the authority itself.

Consider now a similar situation with four agents involved in the loop, with the initial situation as described in Table 3.

Agent ID	Owns	Knows	Preference rank
A_1	R_d	A_2 owns R_c	$R_a > R_b > R_c > R_d$
A_2	R_c	A_3 owns R_b	$R_d > R_a > R_b > R_c$
A_3	R_b	A_4 owns R_a	$R_c > R_d > R_a > R_b$
A_4	R_a	A_1 owns R_d	$R_b > R_c > R_d > R_a$

Table 3: Initial state of the negotiation example with 4 agents

In this case in the first round we have four offers, namely:

- $O_1 = Off(t_0, (A_1, A_2, Exch(R_d, R_c)))$
- $O_2 = Off(t_0, (A_2, A_3, Exch(R_c, R_b)))$
- $O_3 = Off(t_0, (A_3, A_4, Exch(R_b, R_a)))$
- $O_4 = Off(t_0, (A_4, A_1, Exch(R_a, R_d)))$

As in the case above we have consequently four withdrawals, four offers in turn and so on (see the framework AF_{neg4} in Figure 9).

Skipping technical details, it turns out that:

- all three sets representing the reiteration of a specific offer, namely $O_1, O_5, O_9, \ldots$, $O_2, O_6, O_{10}, \ldots$, $O_3, O_7, O_{11}, \ldots$, and $O_4, O_8, O_{12}, \ldots$, are admissible;
- two of the pairwise unions of these sets are admissible namely $O_1, O_3, O_5, \ldots$, and $O_2, O_4, O_6, \ldots$;
- each set consisting of one of the above mentioned pairwise unions and of the withdrawals of the two other offers is stable.

On the basis of these evaluations, it emerges that two exchanges can be executed, with the choice left again to the authority.

In general, using the evaluation of an infinite framework, the authority can go beyond detecting and stopping non terminating situations in this kind of multi-agent dialogues: the added-value consists in identifying which exchanges are anyway feasible in such situations.

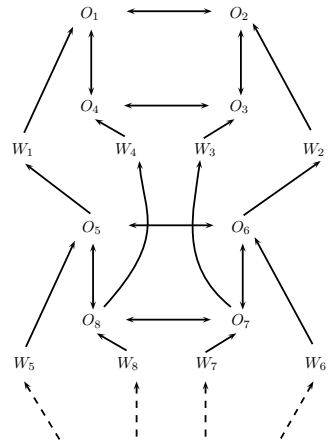

Figure 9: Graphical representation of AF_{neg4}.

7.4. An example in ambient intelligence (from Sec. 2.2.2)

Referring to the description of the example given in Section 2.2.2 and omitting the burden of some uninteresting details (in particular all arguments corresponding to default assumptions which are contradicted by facts), the argumentation framework corresponding to the interactions among the components of the ambient intelligence system consists of:

- a finite part corresponding to basic facts which are not time-dependent, namely $F_1 = person(Brian)$, $F_2 = room(office)$, $F_3 = phone(Brianphone)$, $F_4 = owner(Brianphone, Brian)$ and are not involved in attack relations;
- an infinite part consisting of the regular iteration of a section corresponding to even time instants and a section corresponding to odd time instants[24].

The following arguments and attacks are common to all sections independently of oddness or evenness of the time instant i:

- two facts corresponding to device readings: $NVR(i) = \neg videorecogn(Brian, office, i)$, $PI(i) = phonein(Brianphone, office, i)$;
- an argument $PL(i)$ with conclusion $phlocated(Brianphone, i)$ derived from fact $phonein(Brianphone, office, i)$ using $(r8)$;
- an argument $VV(i)$ with conclusion $videovalid(office, i)$ derived using $(r9)$ on the basis of the default assumption $not\ dark(office, i)$;

[24] A similar but more articulated structure would arise in case the different sensors produce data with different periods.

- an argument $IN(i)$ with conclusion $in(Brian, office, i)$ derived using $(r1)$ on the basis of the default assumption $not\ videovalid(office, i)$;
- an argument $NIN(i)$ with conclusion $\neg in(Brian, office, i)$ derived using $(r3)$ on the basis of the fact $\neg videorecogn(Brian, office, i)$ and of the previously derived conclusion $videovalid(office, i)$;

The following arguments are included only in sections corresponding to an even time instant j:

- the fact $D(j)$ corresponding to the device reading $dark(office, j)$;
- an argument $LO(j)$ with conclusion $lighton(office, j + 1)$ derived using $(r6)$.

The following arguments are included only in sections corresponding to an odd time instant k:

- the fact $ND(k)$ corresponding to the device reading: $\neg dark(office, k)$;
- an argument $NLO(k)$ with conclusion $\neg lighton(office, k+1)$ derived using $(r7)$.

As to attacks:

- each argument with conclusion $videovalid(office, i)$ attacks the argument with conclusion $in(Brian, office, i)$;
- arguments with conclusion $in(Brian, office, i)$ and $\neg in(Brian, office, i)$ mutually attack each other;
- each fact $dark(office, i)$ attacks the arguments with conclusions $videovalid(office, i)$ and $\neg in(Brian, office, i)$.

The corresponding argumentation framework AF_{amb} is depicted in Figure 10. The relevant AF specification can be given as follows. Let $\Sigma = \{F_1, F_2, F_3, F_4, 0\}$ and $\mathcal{X} = \{F_1, F_2, F_3, F_4\} \cup \{0^i \ : \ i \geq 1\}$. We can partition $\mathcal{X} \setminus \{F_1, F_2, F_3, F_4\}$ into 10 sets L_A, L_B, L_C, L_D, L_E, L_F, L_G, L_H, L_I, L_J,(corresponding respectively to the 10 sequences NVR(i), PI(i), PL(i), VV(i), IN(i), NIN(i), D(2i), LO(2i), ND(2i+1), NLO(2i+1), with $i \geq 0$:

$$\begin{array}{lcl} L_A & = & \{0^{8i+1} \ : \ i \geq 0\} \\ L_B & = & \{0^{8i+2} \ : \ i \geq 0\} \\ L_C & = & \{0^{8i+3} \ : \ i \geq 0\} \\ L_D & = & \{0^{8i+4} \ : \ i \geq 0\} \\ L_E & = & \{0^{8i+5} \ : \ i \geq 0\} \\ L_F & = & \{0^{8i+6} \ : \ i \geq 0\} \\ L_G & = & \{0^{16i+7} \ : \ i \geq 0\} \\ L_H & = & \{0^{16i+8} \ : \ i \geq 0\} \\ L_I & = & \{0^{16i+15} \ : \ i \geq 0\} \\ L_J & = & \{0^{16i+16} \ : \ i \geq 0\} \end{array}$$

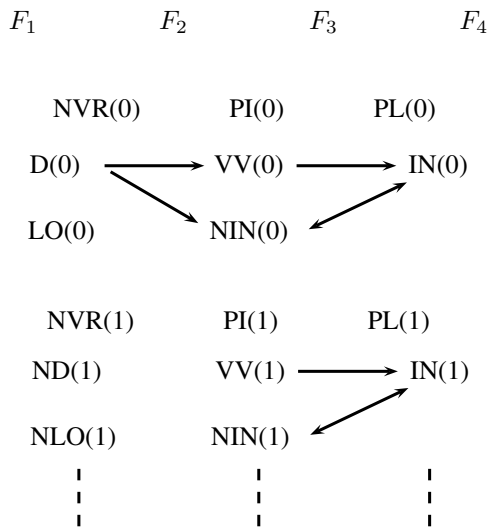

Figure 10: Graphical representation of AF_{amb}.

The attack expression can then be formulated as follows:
$a = tl(I \cap L_E) \cup (((I \cap L_F) \cdot 0) \cap L_G) \cup (((I \cap L_D) \cdot 000) \cap L_G)) \cup ((I \cap L_E) \cdot 0) \cup (tl(I \cap L_F))$.

We observe that the attack expression satisfies the hypothesis of Proposition 9 hence it can be determined that the argumentation framework is finitary. Algorithm 1 can then be applied and it can be verified that it terminates determining the grounded extension $G = \{F_1, F_2, F_3, F_4\} \cup L_A \cup L_B \cup L_C \cup (L_D \cap \{0^{16i+12} \ : \ i \geq 0\}) \cup (L_E \cap \{0^{16i+5} \ : \ i \geq 0\}) \cup (L_F \cap \{0^{16i+14} \ : \ i \geq 0\}) \cup L_G \cup L_H \cup L_I \cup L_J$. Using the method included in the proof of Theorem 3 it can also be verified that G is stable, which implies that G is also the unique preferred extension.

From the argumentation perspective the situation is not pahological *per se* and in fact this oscillating behavior is the desired one in case a person continuously enters and exits a room. Computing a compact representation of the grounded extension is however useful since it can be passed to a higher-level reasoning module which may detect the anomaly that the conclusions entailed by the system involve a person entering and exiting the same room let say every 5 seconds (or less). It can also be observed that, in this case, the produced sequence of arguments is not actually infinite since the oscillating behavior will stop with the sunrise the morning after (or the semester after if we are in a polar winter). However, we are interested in analyzing (and stopping) such a very long sequence of arguments produced with a regular pattern well before it reaches its "natural" termination. To this purpose it can be definitely more advantageous to treat it as an infinite sequence with compact representation rather than dealing explicitly with a finite sequence of thousands (if not millions) of "machine-produced-always-the-same" arguments.

8. Related Work

Treatments of infinite AFs have, as already outlined, been largely limited to specific instances exemplifying particular properties, e.g. that infinitary frameworks may occur naturally, as in the main example from [37] presented in the previous section, or the issue of existence of semi-stable extensions [30, 91]. Beyond such examples the principal results have not advanced noticeably since the general properties proven in [37] were established. In particular, the question of *computational* issues in infinite AFs has not been considered.

At heart (interpretative matters aside) Dung's AF model is graph-theoretic (a property exploited in much extant work on algorithmic and complexity treatments of AFs). The computational theory of infinite directed graphs has, in contrast, long been recognised as a core area of graph theory, arguably dating back to the beginning of the 20th century in the work of Thue [86]. Indeed, as observed by Morvan [66]: "When dealing with computers, infinite graphs are natural objects".

The idea of viewing vertex sets as a formal language with an edge relationship determined by operations on words representing vertices dates back at least as far as Muller and Schupp [67] and much of the focus of such computational treatments from a graph-theoretic perspective has tended to concentrate on, what may loosely be termed, "specification processes" for generating families of infinite graphs and model-theoretic treatments of logics defined via these processes. Thus, Courcelle [33] addresses properties expressible in monadic second-order logic with respect to bounded-width infinite graphs; Blumensath and Grädel [19] consider model-theoretic issues for properties expressible in first-order logic augmented with a quantifier, $\exists^{\omega}$, expressing the existence of infinitely many objects within its scope. The "reachability problem" (given u and v is there a directed path of edges from u to v) of importance in analyses of program behaviour, has been widely studied, e.g in Thomas [85] and Colcombet [31].

In these treatments, as well as in our own approach, the central concern is that of "finite presentations of infinite objects" and so, unsurprisingly, the mechanisms adopted exhibit some structural similarities, e.g. in the use of automata-theoretic models. Overall, however, the issues of interest differ: in particular, aside from specialised studies such as that of Bean [13] regarding colourings of infinite graphs, properties impinging directly on graph-theoretic views of extension-based semantics have not explicitly been dealt with.

Turning to another field related to argumentation, infinite structures have also received a significant deal of attention in the field of logic programming where admitting function symbols and recursion in the language gives rise to possibly infinite domains. Hence, a significant gain in expressiveness has to be traded off with the possibility of actual implementation in practical solvers. Focusing on the family of ASP (Answer Set Programming) solvers, Bonatti [21, 22] investigated the class of *finitary* logic programs which admit unbounded (possibly infinite) domains and cyclic definitions while ensuring that inference is r.e.-complete. Finitary logic programs are therefore amenable to implementation within existing ASP solvers with suitable extensions. A larger class of logic

programs with functions called *finitely grounded* is shown to preserve most of the good properties of finitary programs in [12]. Unfortunately the class of *finitary* logic programs is undecidable: several subsequent works have then been devoted to investigate other classes of logic programs allowing functions, trading off expressiveness and tractability in various ways. In [82, 48] a decidable class of disjunctive logic programs with function symbols under stable model semantics, called $\mathbb{FDNC}$, is introduced and a method is provided to finitely represent all the (possibly infinite) stable models of a given $\mathbb{FDNC}$ program. In [26, 28] the semi-decidable class of *finitely ground* programs is considered, along with its decidable subclass of *finite domain* programs, while another decidable subclass, called *argument restricted*, has been analyzed in [60]. Further, a decidable subclass of finitary programs, called *FP2*, has been recently presented in [11]. On the implementation side, the DLV solver [59] has been extended to encompass the treatment of finitely ground and finite domain programs resulting in a publicly available system called DLV-complex [27].

The above studies witness a large interest in reasoning with infinite domains in answer set programming, with a range of motivations including the explicit treatment of recursive data structures like lists and trees, the encoding of problems not admitting a priori bounds on the solution size (e.g. planning or reasoning about actions), and the consideration of potentially infinite processes in time (a biology-inspired example is provided in [48]). While many of the above needs are common to argumentation theory (and more generally to any approach to defeasible reasoning, as remarked in Section 2) it has to be acknowledged that the significant advancements both on the theoretical and on the application side surveyed above have no counterpart (yet) in the argumentation field, so that the useful connections and interplay between the two fields have definitely to be regarded as a future research subject. As far as the present work is concerned, it can be remarked in particular that the investigations surveyed above lie at the level of the representation language, which is abstracted away in Dung's framework, hence our work concerns a different, and not directly comparable, abstraction level. Moreover the above works are based on the stable model semantics adopted in the context of ASP solvers, while the approach proposed in this paper is not committed to a specific semantics choice and hence is applicable beyond the limits of the stable semantics, which, as well-known, does not always guarantees the existence of extensions (the existence of models in the logic programming context) and does not feature, in general, some desirable properties like directionality or relevance (see [8] for a discussion).

9. Further Work and Conclusions

Our main aim in this paper has been to present a formal approach to describe both finite and infinite AF structures, the argument set being the set of words within some regular language, $\mathcal{X} \subseteq \Sigma^*$, and the attack relation, $\mathcal{A}$ over $\mathcal{X} \times \mathcal{X}$ being given through a sentence, $a \in \mathcal{AE}(\Sigma)$ constructed by a limited set of operations so that for $S \subseteq \mathcal{X}$, $\underline{a}(S)$ satisfies additivity (hence also monotonicity) and preserves regularity. We provided some illustrations of the flexibility of our

approach using examples from [37, 30]. More generally, the approach has been shown to be able to capture standard finite AFs and arbitrary finite combinations of finite and infinite AFs, which can reasonably be regarded as covering most (if not all) situations of practical interest.

A related research line we are developing in parallel concerns the use of this kind of techniques to represent infinite structures in extended versions of Dung's framework, some initial results concerning the AFRA formalism (Argumentation Framework with Recursive Attacks) having been recently obtained [4].

We have concentrated on the expressive potential of AFS, indicating that, in contrast to "naive" encodings, processes which can be dealt with efficiently in the finite setting – deciding conflict-freeness, admissibility, acceptability, verifying whether a set is a stable or complete extension as well as construction problems such as computing the characteristic function – all admit effective decision methods and algorithms for building automata accepting the corresponding sets, even when the instances being checked or the results reported are themselves infinite subsets of $\mathcal{X}$. For the case of two problems, – existence of stable extensions and determining credulous acceptance wrt preferred semantics – unlikely to be efficiently decidable in the finite context we have shown that within AFS these are (at worst) semi-decidable.

We conclude by reviewing some topics meriting further development, a number of which are the subject of current work. One such immediate area of interest concerns the *efficiency* with which particular procedures can be implemented (as opposed to the issue of *effectiveness*). While some preliminary study of such questions is underway we have chosen, partly for reasons of space, not to develop this aspect in detail within the current paper. We note that such questions concern two elements: the *size* (i.e. number of states) of automata achieving particular tasks, and the computational complexity of the problems themselves. The former, referred to as *state complexity* in the associated literature has been widely studied[25] so that tight bounds on state complexity delineating the number of states necessary and sufficient for an automaton accepting $R\theta S$ or $\theta(R)$, in terms of the state complexity of the languages R and S have been obtained for each of the principal operations θ with each of the finite automaton forms discussed. It is, clearly, the case that the extent to which, say, $\pi_a^-(S)$, may be recognised by a "small" automaton will depend significantly not only on the state complexity of S itself, but also on the exact specification of $a \in \mathcal{AE}(\Sigma)$. As such it would seem unlikely that a completely general treatment of state complexity for $\mathcal{AE}(\Sigma)$ (even if such is possible) will yield results of much interest since this generality is likely to overestimate state complexity for those cases that might arise in practice. A rather more promising approach is to consider sub-classes of $\mathcal{AE}(\Sigma)$ obtained by constraining the operational structures, e.g. given some *finite* "base language", $B \subseteq \Sigma^*$ consider attack structures, $\underline{a}$ satisfying "$v \rightarrow_a w$ only if $v \in B$ or $r(|v|, |w|)$" (so that $r(...)$ is determined through some aspect of the lengths of v and w). In fact preliminary results of the authors,

[25] Important contributions may be found in [65, 18, 58, 71, 92].

with $B \subseteq \Sigma^2$ and the constraint "$v \rightarrow_a w$ iff ($v \in B$ and $tl(B) = hd(w)$) or ($v \in \Sigma \cdot w$)" indicate, using a careful treatment of the DFA form accepting $\mathcal{X}$ that all of the cases shown to be effectively computable in Thm. 3 may be efficiently implemented (in terms of state complexity and polynomial run-time).[26]

A further topic of some interest concerns the use of our approach in *finite* frameworks. Although it is, of course, unnecessary to resort to AFS schema to describe finite $\langle \mathcal{X}, \mathcal{A} \rangle$ there are, however, cases where it may be advantageous to do so. For example, suppose $|\mathcal{X}| = 2^m$ for some $m \in \mathbf{N}$, then using $\Sigma = \{0, 1\}$ the set $\mathcal{X}$ can be viewed as $\{0, 1\}^m$ a language that is accepted by a DFA with exactly $m + 1$ states. Thus, for suitable $\mathcal{A} \subseteq \mathcal{X} \times \mathcal{X}$ the AF, $\langle \mathcal{X}, \mathcal{A} \rangle$ rather than requiring a description whose size is $O(2^m) + |\mathcal{A}|$ could be presented by one whose size is $O(m^k)$ for some $k \in \mathbf{N}$. In cases where such compaction can be achieved, an important issue is the resulting cost of implementing standard decision procedures: an obvious concern is that, for this particular finite case, some subsets of $\{0, 1\}^m$ will require automata whose state complexity is $2^{O(m)}$. It is, however, unclear whether this behaviour would be the *only* potential drawback, e.g. what can be said regarding the complexity of CA_{adm} (for single, rather than sets of arguments) in such settings?

As a final collection of problems we note that several issues remain open concerning *effective* decision processes for extension-based semantics in AFs. In particular, although we have shown questions such as $\mathcal{E}_{stab} = \emptyset$ to be semi-decidable (in finitary AFs), the status of its converse is open, i.e. is it the case that $\mathcal{E}_{stab} \neq \emptyset$ is semi-decidable? A positive answer would, of course, lead to an effective procedure for EXIST_{stab}, while a negative answer motivates the question of identifying decidable fragments of $\mathcal{AE}(\Sigma)$.

On a different side, it has to be acknowledged that AFS is not an immediately usable formal tool and that the specification of each example has been crafted individually. In perspective, AFS can be regarded as a "low level" language which can represent the basis for the definition of higher level constructs for the description of infinite AFs, accompanied by suitable methodologies for their application. In fact, some recurrent structural and representation patterns can be identified in the examples considered in the paper and procedures to derive AFS from logic programs could be considered, but a full investigation of these issues is left for future work.

In conclusion we emphasise once more that the development put forward in this paper, while establishing many cases where an effective treatment of infinite argumentation forms is realistic, provides a starting point for a wider investigation of this matter.

[26] We remark that the AFRA structures described in [4] are a special case of this restriction: illustrative efficient automata constructions (in terms of both state complexity and algorithm run-time) have been obtained.

Appendix A. Formal Languages and Automata

A standard approach to the problem of representing an infinite collection of objects via a finite specification is to exploit so-called formal grammars and their associated machine models. In this section we review some basic elements and results from this discipline to complement the basic definitions given in Section 3.2.

Given a formal grammar, $G = \langle \Sigma, V, P, S \rangle$ (Definition 8), and a production rule $\alpha \to \beta \in P$, for all $\gamma,\ \delta \in (V \cup \Sigma)^*$ we say that $\gamma\alpha\delta$ *derives* $\gamma\beta\delta$ in G ($\gamma\alpha\delta \Rightarrow_G \gamma\beta\delta$) and in general $u \Rightarrow_G^* w$ whenever there is a finite sequence of derivations such that

$$u \Rightarrow_G u_1 \Rightarrow_G u_2 \Rightarrow_G \cdots \Rightarrow_G u_k \Rightarrow_G w$$

A derivation $u \Rightarrow_G w$ is *terminated* if $w \in \Sigma^*$. The *language generated* by $G = \langle \Sigma, V, P, S \rangle$, denoted as $L(G)$, is

$$L(G) \quad = \quad \{\ w \in \Sigma^*\ :\ S \Rightarrow_G^* w\}$$

A language, $L \subseteq \Sigma^*$, is *recognisable* if there is a formal grammar $G = \langle \Sigma, V, P, S \rangle$ for which $w \in L$ if and only if $w \in L(G)$.

Notice that, in general, formal grammars provide a process for proving that $w \in L(G)$ and that there is not, necessarily, a *unique* sequence of derivations under which $S \Rightarrow_G^* w$.

Definition 17. A grammar $\langle \Sigma, V, P, S \rangle$ is *unrestricted* if P is allowed to contain arbitrary rules $\alpha \to \beta$ (subject to the constraint that $\alpha \notin \Sigma^*$). It is *context–sensitive* if $\forall\ \alpha \to \beta \in P$ we have $|\beta| \geq |\alpha|$; *context–free* if $\forall\ \alpha \to \beta \in P$ we have $\alpha \in V$ and *right–linear* if every $\alpha \to \beta \in P$ has the form $V_i \to \varepsilon$ or $V_i \to \sigma$ or $V_i \to \sigma V_j$ for $V_i,\ V_j \in V$ and $\sigma \in \Sigma$.

Recall that a language L is *recursively enumerable* (r.e.) if there is a Turing machine (TM) program, M, that given any $w \in L$ as input will eventually halt and accept w; with L being *recursive* if there is a TM program, M, that given any $w \in \Sigma^*$ as input eventually halts and accepts any $w \in L$ and halts and rejects any $w \notin L$. We use the term *decidable* to describe languages which are recursive and *semi-decidable* for those which are recursively enumerable. The term *effective algorithm* for L will be used for an algorithmic process, e.g. a Turing machine program, that witnesses L as decidable.[27]

Fact 3.

a. $L \subseteq \Sigma^*$ is *recursively enumerable* if and only if there is an unrestricted grammar, G such that $L(G) = L$.

[27] It should be noted that, some closure properties are established non-constructively so that effective algorithms yielding machines recognising the resulting language do not necessarily follow, see e.g. [57, pp. 62–63].

b. $L \subseteq \Sigma^*$ is *recursive* if and only if there are unrestricted grammars, G_1 and G_2 such that $L(G_1) = L$ and $L(G_2) = \overline{L}$, i.e. $L(G_2) = \Sigma^* \setminus L$.

It is well known that there are languages that fail to be r.e.

Regular languages (Definition 7) are captured by a syntactic formalism called *regular expressions*. A regular expression, E over Σ is constructed by a finite number of applications of the following

$$\begin{cases} \emptyset & \text{is a regular expression} \\ \varepsilon & \text{is a regular expression} \\ \sigma & \forall\ \sigma \in \Sigma \text{ is a regular expression} \\ (R + S) & \text{for regular expressions } R,\ S \\ R \cdot S & \text{for regular expressions } R,\ S \\ (R)^* & \text{for a regular expression } R \end{cases}$$

The associated regular languages $L(E) \subseteq \Sigma^*$ being,

$$\begin{cases} \emptyset & \text{if } E = \emptyset \\ \{\varepsilon\} & \text{if } E = \varepsilon \\ \{\sigma\} & \text{if } E = \sigma \\ L(R) \cup L(S) & \text{if } E = (R + S) \\ L(R) \cdot L(S) & \text{if } E = R \cdot S \\ L(R)^* & \text{if } E = (R)^* \end{cases}$$

In order to reduce notational complications we will, in general, equate a regular expression, R, with the language, $L(R)$, it describes, thus writing R for both cases. Where no ambiguity arises, we dispense with superflous parentheses.

Fact 4. Let REG $\subseteq 2^{\Sigma^*}$ be the property describing all regular languages, i.e. $L \in$ REG if and only if L is a regular language. The class REG is closed with respect to *all* of the operations $\theta \in \{\cup,\ \cap, \overline{\{\ \}}, \setminus, \cdot, ^*, /, rev\}$.

The class of machine models that express exactly the regular languages are the *deterministic finite automata* (Definition 9), other classes of finite automata can also be considered.

Definition 18. A *non-deterministic finite automaton* (NDFA) $NM = \langle \Sigma, Q, q_0, F, \delta \rangle$ has $\delta\ :\ Q \times \Sigma \to 2^Q$, indicating that in some states and symbols there may be more than one "next" state (or even that no state at all can be reached should $\delta(q, \sigma) = \emptyset$). An ε-NDFA has a state transition function $\delta\ :\ Q \times \Sigma \cup \{\varepsilon\} \to 2^Q$ where the interpretation of $\delta(q, \varepsilon) = Q' \subseteq Q$ is that having reached state q the automaton may process its next input symbol $\sigma \in \Sigma$ from q itself *or* from any state in $\delta(q, \varepsilon)$. We identify a sub-class, the so-called "ε–DFA" of ε-NDFA via those whose transition function satisfies: $\delta\ :\ Q \times \Sigma \to Q$ and $\delta\ :\ Q \times \{\varepsilon\} \to 2^Q$, i.e. ε-DFA specify *exactly one* successor state for each $q \in Q$ and $\sigma \in \Sigma$ but can allow arbitrary ε transitions between states.

For a NDFA, M, $w = w_k \cdot w_2 \cdots w_1 \in \Sigma^*$ is accepted by M, written $w \in L(M)$ if there is *at least one* sequence of states $q_{i_1} q_{i_2} \ldots q_{i_k}$ such that $q_{i_1} \in \delta(q_0, w_1)$, $q_{i_j} \in \delta(q_{i_{j-1}}, w_j)$ for $2 \leq j \leq k$ and $q_{i_k} \in F$. For ε-NDFA $w = w_k \cdot w_{k-1} \cdots w_1 \in$

Σ^* is accepted by M, if there is a *finite* sequence $q_{j_1} q_{j_2} \dots q_{j_r}$ of states with $r \geq k$ and a finite sequence $\alpha_1 \alpha_2 \dots \alpha_r$ with $\alpha_i \in \{\varepsilon\} \cup \Sigma$ such that: $\alpha_r \cdot \alpha_{r-1} \cdots \alpha_1 = w$, $q_{i_1} \in \delta(q_0, \alpha_1)$, $q_{i_j} \in \delta(q_{i_{j-1}}, \alpha_j)$ for $2 \leq j \leq r$, and $q_{j_r} \in F$. The concept of acceptance by ε-DFA is defined similarly.

Fact 5. For $L \subseteq \Sigma^*$ the following are equivalent.

a. L is a regular language.
b. There is an ε-NDFA, M, for which $L(M) = L$.
c. There is an ε-DFA, M, for which $L(M) = L$.
d. There is a NDFA, M, with $L(M) = L$
e. There is a DFA, M with $L(M) = L$.
f. There is a right-linear grammar, G, for which $L(G) = L$.

Fact 6.

a. Given any finite automaton (DFA, NDFA, ε-DFA or ε-NDFA), $\mathcal{M} = \langle \Sigma, Q, q_0, F, \delta \rangle$, it may decided in polynomial time (in $|Q| + |\Sigma|$) if $L(\mathcal{M}) = \emptyset$.
b. Given two DFAs accepting languages L_1 and L_2 there are effective algorithms for constructing a DFA accepting $L_1 \cap L_2$, $L_1 \cup L_2$, $L_1 \setminus L_2$, L_1 / L_2.
c. Every regular language $L \subseteq \Sigma^*$ has a *unique*[28] minimal number of states DFA, M for which $L(M) = L$. Furthermore, given M' with $L(M') = L$ the unique minimized automaton, M with $L(M) = L(M') = L$ may be constructed in polynomial time in $|Q_{M'}| + |\Sigma|$.

Fact 6 (c) is the Myhill-Nerode Theorem [68], a polynomial time algorithm for constructing the minimal automaton may be found in [57, Ch. 3.4]; the most efficient (currently known) algorithm is that of Hopcroft [56] which takes at most $O(|Q| \log |Q|)$ steps to minimise a DFA with $|Q|$ states.

Appendix B. Proofs

Appendix B.1. Proofs of Section 4

Proposition 1. For $\mathcal{X}$ as introduced in Definition 10, there are choices of $\mathcal{A} \subseteq \mathcal{X} \times \mathcal{X}$ such that there is no formal grammar, G with $L(G) = L_{\mathcal{A}} \subseteq \{0^i \cdot 1 \cdot 0^j \ : \ i, \ j \geq 1\}$.

Proof. It is well-known that Turing machine programs may be encoded as words in $\{0,1\}^*$ in such a way that if $\beta(M)$ is the encoding of some TM, M, then there is a (so-called *universal*) TM which given the pair $\langle \beta(M), w \rangle$ as input, exactly simulates the computational steps of M on input w.[29] Furthermore it can be decided if any $w \in \{0,1\}^*$ is such that $w = \beta(M)$ for some TM program M. For

[28] "Uniqueness" is modulo relabelling states of the automaton.
[29] See, for example, Hopcroft and Ullman [57, Chap. 8.3] or Dunne [38, Chap. 4] for example constructions of such universal TMs.

any such encoding scheme we may use the standard *lexicographic ordering*[30] of $\{0,1\}^*$ to order TM programs, so that

$$\beta(M_i) \quad = \quad \text{The } i\text{'th word in the lexicographic ordering of } \{0,1\}^* \text{ that describes a valid TM encoding.}$$

Finally, we recall that there is no formal grammar, G, that generates the following language:

$$\text{NON-HALT-EMPTY} \;=\; \{\beta(M) \;:\; M \text{ does } \textit{not} \text{ halt given } \varepsilon \text{ as input}\}$$

We can now define the language $L_{\mathcal{A}}$ with the property required via

$$L_{\mathcal{A}} \quad = \quad \{0 \cdot 1 \cdot 0^k \;:\; \beta(M_k) \in \text{NON-HALT-EMPTY}\}$$

From which it follows that a grammar G with $L(G) = L_{\mathcal{A}}$ allows a grammar $G_{\neg\varepsilon - halt}$ with $L(G_{\neg\varepsilon - halt}) = \text{NON-HALT-EMPTY}$ to be built. □

Proposition 2. Given an arbitrary (i.e. unrestricted) grammar G over the alphabet $\{0,1\}$ the problem of determining if $L(G) \subseteq \{0^i \cdot 1 \cdot 0^j \;:\; i,\; j \geq 1\}$ is not semi-decidable, i.e. there is no TM program which given (a description of) G as input halts and accepts precisely those G for which $L(G) \subseteq \{0^i \cdot 1 \cdot 0^j \;:\; i,\; j \geq 1\}$.

Proof. Immediate consequence of Rice's Theorem for r.e. Index Sets, [79], see e.g. [57, pp. 189–192] or [38, pp. 57–66].[31] □

Proposition 3. Given an arbitrary *context-sensitive* grammar, G, over the alphabet $\{0,1\}$ the problem of determining if $L(G) \subseteq \{0^i \cdot 1 \cdot 0^j \;:\; i,\; j \geq 1\}$ is not semi-decidable.

Proof. The problem of determining if $L(G) = \emptyset$ for an arbitrary context-sensitive grammar (over alphabet $\{0,1\}$) is not semi-decidable. Given a context-sensitive grammar G over $\{0,1\}$ we construct a context-sensitive grammar G' over $\{0,1\}$ with the property that $L(G') \subseteq \{0^i \cdot 1 \cdot 0^j \;:\; i,\; j \geq 1\}$ if and only if $L(G) = \emptyset$. Let S be the start symbol of G. Add a new starting symbol S' to G with a single production $S' \to 1 \cdot S$ to give the new grammar G'. Then given that any word actually generated by G' must begin with the symbol 1, the only way in which $L(G') \subseteq \{0^i \cdot 1 \cdot 0^j \;:\; i,\; j \geq 1\}$ would be if $L(G') = \emptyset$. This can only be the case if $L(G) = \emptyset$ to begin with. □

Proposition 4. Given an arbitrary *context-free* grammar (CFG), G, over the alphabet $\{0,1\}$ the problem of determining if $L(G) \subseteq \{0^i \cdot 1 \cdot 0^j \;:\; i,\; j \geq 1\}$ is decidable.

---

[30] That is, the total ordering $<_{\text{lex}}$ in which $0 <_{\text{lex}} 1$, $w <_{\text{lex}} u$ if $|w| < |u|$, and, when $|w| = |u|$ $w <_{\text{lex}} u$ if $w = 0 \cdot v$, $u = 1 \cdot x$ or (when $w = \alpha \cdot v$ and $u = \alpha \cdot x$) if $v <_{\text{lex}} x$.

[31] Rice's Theorem for r.e. Index Sets characterises those "properties" of TMs (equivalently, formal grammars) that are semi-decidable. It is trivial to show that grammars generating subsets of $\{0^i \cdot 1 \cdot 0^j \;:\; i,\; j \geq 1\}$ fail to meet the conditions for a property to be semi-decidable.

Proof. First note that $\underline{L_{010}} = \{0^i \cdot 1 \cdot 0^j \ : \ i,\ j \geq 1\}$ is a regular language, and hence its complement $\overline{L_{010}}$ is a regular language too. Now, given G a context-free grammar over the alphabet $\{0,1\}$, checking $L(G) \subseteq L_{010}$ is equivalent to check $L(G) \cap \overline{L_{010}} = \emptyset$. It is well-known [57] that the intersection of a context-free language (in our case $L(G)$) with a regular language (in our case $\overline{L_{010}}$) is a context-free language, whose specification can be constructed from those of $L(G)$ and $\overline{L_{010}}$. The conclusion then follows from the fact that verifying the emptyness of the language generated by a context-free grammar can be done in polynomial time [57]. □

Proposition 5. Given $M = \langle Q, \{0,1\}, q_0, F, \delta\rangle$ a DFA over the alphabet $\{0,1\}$ there is a polynomial (in $|Q|$) algorithm that decides $L(M) \subseteq \{0^i \cdot 1 \cdot 0^j \ : \ i,\ j \geq 1\}$.

Proof. The DFA, M, accepts a subset of $\{0^i \cdot 1 \cdot 0^j \ : \ i,\ j \geq 1\}$ if and only if $L(M) \setminus \{0^i \cdot 1 \cdot 0^j \ : \ i,\ j \geq 1\} = \emptyset$. Noting the language $\{0^i \cdot 1 \cdot 0^j \ : \ i,\ j \geq 1\}$ is regular and that for DFAs, M, M' a DFA accepting exactly $L(M) \setminus L(M')$ may be constructed in polynomial time, the proof is completed by observing that $L(M) = \emptyset$ is also decidable in polynomial time for any given DFA. □

Proposition 6. Let L be any subset of $\{0^i \cdot 1 \cdot 0^j \ : \ i,\ j \geq 1\}$ with the following property: there are infinitely many values of k such that $\{0^k \cdot 1 \cdot 0^m \ : \ m \geq 1\} \cap L \neq \emptyset$ *and* for all $0^n \cdot 1 \cdot 0^m \in L$, $n \leq m$. Then L is *not* a regular language.

Proof. From the Pumping Lemma for regular languages, cf. [57, Chap. 3.1], with any regular language, L, there is an associated constant, K_L, such that: for all $w \in L$, with $|w| \geq K_L$, $w = x \cdot y \cdot z$ with $|x \cdot y| < K_L$, $|y| \geq 1$ and $x \cdot y^t \cdot z \in L$ for all $t \geq 0$. Thus proceeding by contradiction it suffices to consider some $w = 0^r \cdot 1 \cdot 0^s \in L$ with $r \geq K_L$: note that the existence of a suitable w is guaranteed by the premise that there are infinitely many distinct values of k for which $0^k \cdot 1 \cdot 0^m \in L$. Now, since by the condition $|x \cdot y| < K_L \leq r$ 1 does not belong to $x \cdot y$, we can write $w = x \cdot y \cdot 0^a \cdot 1 \cdot 0^s = 0^p \cdot 0^q \cdot 1 \cdot 0^s$ with $q = |y|$, $p = |x| + a$, and $p + q < K_L$. It follows that all words of the form $0^{p+tq} \cdot 1 \cdot 0^s$ are in L for all $t \geq 0$. Now choosing t so that $p + tq > s$ yields a word which violates the conditions for membership in L. □

Appendix B.2. Proofs of Section 5

Theorem 1. Let $p \in \mathcal{AE}(\Sigma)$ be *any* attack expression over Σ. The mapping $\underline{p} \ : \ 2^{\Sigma^*} \to 2^{\Sigma^*}$ is reasonable.

Proof. Let p be an attack expression over Σ. We proceed by induction on $size(p) \geq 0$.

The inductive base case involves $p \in \{\sigma_1, \ldots, \sigma_k, I\}$. First observe that $\underline{p}$ in these cases satisfies the additivity requirement (R1) of Defn. 12 since for any $S \subseteq \Sigma^*$ we have $\underline{p}(S) \ \in \{\sigma_1, \ldots, \sigma_k, S\}$ and for each $w \in S$, $\underline{p}(\{w\}) \ \in \{\sigma_1, \ldots, \sigma_k, \{w\}\}$. Hence in the case $p = \sigma$ we obtain

$$\underline{p}(S) \ = \ \{\sigma\} \ = \ \bigcup_{w \in S} \{\sigma\} \ = \ \bigcup_{w \in S} \underline{p}(\{w\})$$

whereas for $p = I$ we have

$$\underline{p}(S) \;=\; S \;=\; \bigcup_{w\in S} \{w\} \;=\; \bigcup_{w\in S} \underline{p}(\{w\})$$

Finally, since S is assumed regular to begin with, for each of the base case possibilities, we have $\underline{p}(S)$ is also regular.

Now inductively assume for some $k > 0$ and all attack expressions over Σ, q, with $size(q) < k$ the mapping given via $\underline{q}$ is a reasonable attack function. Consider any attack expression, p, over Σ for which $size(p) = k$. Since $size(p) > 0$ its construction must involve (at least) one of the operations from $\{\cup, K_\Sigma\cdot, \cdot K_\Sigma, /K_\Sigma, K_\Sigma/, \cap K_\Sigma, hd, tl\}$. We consider these in turn.

If $p = q \cup r$ then $\underline{p}(S) = (\underline{q}(S) \cup \underline{r}(S))$. By the inductive hypothesis $\underline{q}$ and $\underline{r}$ are both reasonable, hence since $\cup$ preserves both the properties (R1) and (R2) it follows that $\underline{p}$ is reasonable.

If $p = K_\Sigma \cdot q$ for some regular subset K_Σ of Σ^*

$$\underline{p}(S) \;=\; K_\Sigma \cdot \underline{q}(S) \;=\; \left(\bigcup_{u\in K_\Sigma} u \cdot \underline{q}(S) \right)$$

where, from the inductive hypothesis, $\underline{q}$ is reasonable. Thus

$$\bigcup_{w\in S} \underline{p}(\{w\}) \;=\; \bigcup_{w\in S} \left(\bigcup_{u\in K_\Sigma} u \cdot \underline{q}(\{w\}) \right)$$

which is

$$\left(\bigcup_{u\in K_\Sigma} u \cdot \underline{q}(S) \right)$$

by the additivity of $\underline{q}$. Again (R2) holds by virtue of the fact that $K_\Sigma\cdot$ preserves regularity.

The argument for $p = q \cdot K_\Sigma$ is similar.

If $p = q/K_\Sigma$ then: $\underline{p}(S) \;=\; \underline{q}(S)/K_\Sigma$, which, by the additivity of $\underline{q}$, is equivalent to

$$\left(\bigcup_{w\in S} q(\{w\}) \right) /K_\Sigma \;=\; \bigcup_{w\in S} q(\{w\})/K_\Sigma \;=\; \bigcup_{w\in S} \underline{p}(\{w\})$$

so that again $\underline{p}$ is additive from the fact that $\underline{q}$ is additive. The closure property is again easily verified.

The case $p = K_\Sigma/q$ is similar.

If $p = q \cap K_\Sigma$ then $\underline{p}(S)$ is

$$\underline{q}(S) \cap K_\Sigma \;=\; \left(\bigcup_{w\in S} \underline{q}(\{w\}) \right) \;\cap\; K_\Sigma$$

and

$$\left(\bigcup_{w\in S} \underline{p}(\{w\})\right) \;=\; \left(\bigcup_{w\in S} \underline{q}(\{w\}) \;\cap\; K_\Sigma\right) \;=\; \left(\bigcup_{w\in S} \underline{q}(\{w\})\right) \;\cap\; K_\Sigma$$

so that again additivity holds. Closure is trivially established.

For $p = hd(q)$, we get

$$\underline{p}(S) \;=\; hd(\{\; w \;:\; w \in \underline{q}(S)\}) \;=\; \bigcup_{u\in S} hd(\{\; w \;:\; w \in \underline{q}(\{u\})\} \;=\; \bigcup_{u\in S} \underline{p}(\{u\})$$

using additivity of $\underline{q}$ for the second equality.

To see that (R2) holds it suffices to note that (given a regular language S) $hd(\underline{q}(S)) \subseteq \Sigma$ is finite and hence trivially a regular language.

For $p = tl(q)$:

$$\underline{p}(S) \;=\; tl(\{\; w \;:\; w \in \underline{q}(S)\}) \;=\; \{\; u \;:\exists \sigma \in \Sigma \text{ s.t. } \sigma \cdot u \in \underline{q}(S)\} \;=$$

$$\bigcup_{w\in S} \{\; u \;:\; \sigma \cdot u \in \underline{q}(\{w\}) \;\} \;=\; \bigcup_{w\in S} \underline{p}(\{w\})$$

using the additivity of $\underline{q}$ for the third equality.

It remains to show $\underline{p}(S)$ is regular if S is so. Consider a DFA, $\mathcal{M}_q$, accepting $\underline{q}(S)$ – such a DFA being guaranteed by the fact that $\underline{q}(S)$ is regular. In order to build a DFA accepting $tl(\underline{q}(S))$ it suffices to replace its accepting states, $\mathcal{F}_q$ by $\{\; r \;:\; \delta(r,\sigma) \in \mathcal{F}_q\}$. We deduce that $tl(q)$ gives rise to $\underline{p}$ satisfying R2 thus completing the inductive argument. □

Proposition 7. If μ is a reasonable mapping over the domain 2^{Σ^*} then $\mu_{\mathcal{X}}$ is a reasonable mapping over the domain $2^{\mathcal{X}}$.

Proof. Let $\mathcal{X} \subseteq \Sigma^*$ and $\mu \;:\; 2^{\Sigma^*} \to 2^{\Sigma^*}$ be a reasonable mapping. First note that $\mu_{\mathcal{X}} \;:\; 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ is additive since for any $S \subseteq \mathcal{X}$ we have $\mu_{\mathcal{X}}(S)$ equal to

$$\mu(S) \cap \mathcal{X} \;=\; \left(\bigcup_{w\in S} \mu(\{w\})\right) \cap \mathcal{X} \;=\; \bigcup_{w\in S} \mu(\{w\}) \cap \mathcal{X} \;=\; \bigcup_{w\in S} \mu_{\mathcal{X}}(\{w\})$$

Finally, that $\mu_{\mathcal{X}}$ preserves regularity for regular subsets S of $\mathcal{X}$ is immediate from $\mu_{\mathcal{X}}(S) = \mu(S) \cap \mathcal{X}$. □

Theorem 2. Let $\langle \mathcal{M}, a\rangle$ be an AFS with $L(\mathcal{M}) = \mathcal{X}$ and $a \in \mathcal{AE}(\Sigma)$. The mapping $\underline{a}^+ : 2^{\Sigma^*} \to 2^{\Sigma^*}$, defined as $\underline{a}^+(S) = \{\; v \in \Sigma^* \;:\; \exists\, u \in S \text{ s.t. } u \in \underline{a}(v)\}$ is closed wrt regular languages.

Proof. Consider the various forms that $a \in \mathcal{AE}(\Sigma)$ may have. We show by induction on $size(a)$ that if S is a regular language then $\underline{a}^+(S)$ is a regular language too.

Case 1. $size(a) = 0$ (Inductive base - Fact 2.1)

In this case, $a \in \{\sigma_1, \ldots, \sigma_k, I\}$.

For $a = \sigma_i$,

$$\underline{a}^+(S) \quad = \quad \begin{cases} \Sigma^* & \text{if} \quad \sigma_i \in S \\ \emptyset & \text{if} \quad \sigma_i \notin S \end{cases}$$

We note, in view of the properties $tl(\emptyset) = \emptyset$, $tl(\{\sigma\}) = \varepsilon$ and $\emptyset \cdot L = \emptyset$, that $\underline{a}^+(S) = tl(S \cap \{\sigma_i\}) \cdot \Sigma^*$, i.e. we do not need to explicitly represent the conditional behaviour, thus allowing one to express a^+ as $tl(I \cap \sigma_i) \cdot \Sigma^*$.

For $a = I$, $\underline{a}^+(S) = S$. Thus in each case $\underline{a}^+(S)$ is a regular language.

Assuming for each a with $size(a) < k$, that $\underline{a}^+(S)$ is regular, consider $a \in \mathcal{AE}(\Sigma)$ with $size(a) = k$.

Case 2.1 $a = b \cup c$ (Fact 2.2)

Then

$$\begin{array}{rcl} \underline{a}^+(S) & = & \{\ v \in \Sigma^* \ :\ \exists u \in S \ \text{ s.t. } u \in \underline{b}(\{v\}) \text{ or } u \in \underline{c}(\{v\})\} \\ & = & \{\ v \in \Sigma^* \ :\ \exists u \in S \ \text{ s.t. } u \in \underline{b}(\{v\})\} \ \cup \\ & & \{\ v \in \Sigma^* \ :\ \exists u \in S \ \text{ s.t. } u \in \underline{c}(\{v\})\} \\ & = & \underline{b}^+(S) \cup \underline{c}^+(S) \end{array}$$

Via the inductive hypothesis and the closure properties of regular languages (see Fact 4 in Appendix A), this is a regular language.

Case 2.2 $a = b \cdot K_\Sigma$ (Fact 2.3)

$$\underline{a}^+(S) \quad = \quad \{\ v \in \Sigma^* \ :\ \exists u \in S \text{ s.t. } u \in \underline{b}(\{v\}) \cdot K_\Sigma\}$$

Recall that the *quotient* of a language L_1 wrt L_2 (denoted L_1/L_2) is

$$L_1/L_2 \quad = \quad \{\ p \ :\ \exists\ q \in L_2 \ \text{ s.t. } p \cdot q \in L_1\}$$

It is easily seen that for $a = b \cdot K_\Sigma$ this leads to

$$\begin{array}{rcl} \underline{a}^+(S) & = & \{\ v \in \Sigma^* \ :\ \exists\ p \in S/K_\Sigma \ \text{ s.t. } p \in \underline{b}(\{v\})\} \\ & = & \underline{b}^+(S/K_\Sigma) \end{array}$$

That is, unless $u \in S$ has the form $p \cdot q$ with $q \in K_\Sigma$, then $\underline{a}^+(\{u\}) = \emptyset$; for $u \in S$ which is of the required form, it is necessary to identify which arguments these (with the K_Σ component removed, i.e. replacing $p \cdot q$, $q \in K_\Sigma$ with p) attack according to the specification b.

Again, this case is completed by recalling that regular languages – which S and K_Σ are by definition - are closed under the quotient operator (see Fact 4 in Appendix A) and the inductive hypothesis which ensures that $\underline{b}^+$ preserves regularity.

Case 2.3 $a = K_\Sigma \cdot b$ (Fact 2.4)

The argument is similar to that used in Case 2.2, so that:

$$\underline{a}^+(S) = \{\ v \in \Sigma^* \ :\ \exists\ u \in S \ \text{ s.t. } u \in K_\Sigma \cdot \underline{b}(\{v\})\}$$

Hence if $w \in \Sigma^*$ does not have the form $p \cdot q$ for some $p \in K_\Sigma$ then $\underline{a}^+(\{w\}) = \emptyset$ otherwise $\underline{a}^+(\{w\}) = \underline{b}^+(\{q\})$, i.e.

$$\underline{a}^+(S) \quad = \quad \underline{b}^+(\ \{\ q \ :\ \exists\ p \in K_\Sigma \text{ s.t. } p \cdot q \in S\}\ )$$

However, noting that $rev(L \cdot R) = rev(R) \cdot rev(L)$, it follows that $p \in K_\Sigma$ and $p \cdot q \in S$ if and only if $rev(q) \cdot rev(p) \in rev(S)$ and $rev(p) \in rev(K_\Sigma)$ so that $\{\ q\ :\ \exists\ p \in K_\Sigma$ s.t. $p \cdot q \in S\})$ is $rev(T)$ where

$$\begin{array}{rcl} T & = & \{\ q \in \Sigma^*\ :\ \exists\ p \in rev(K_\Sigma) \text{ s.t. } q \cdot p \in rev(S)\ \} \\ & = & rev(S)/rev(K_\Sigma) \end{array}$$

and $\underline{a}^+(S) = \underline{b}^+(rev(rev(S)/rev(K_\Sigma)))$ with this case following since regular languages are closed under $rev()$.

Case 2.4 $a = b/K_\Sigma$ (Fact 2.5)

$$\begin{array}{rcl} \underline{a}^+(S) & = & \{\ v \in \Sigma^*\ :\ \exists u \in S \text{ s.t. } u \in \underline{b}(\{v\})/K_\Sigma\} \\ & = & \underline{b}^+(S \cdot K_\Sigma) \end{array}$$

Case 2.5 $a = K_\Sigma/b$ (Fact 2.6)

$$\begin{array}{rcl} \underline{a}^+(S) & = & \{\ v \in \Sigma^*\ :\ \exists u \in S \text{ s.t. } u \in K_\Sigma/\underline{b}(\{v\})\} \\ & = & \{\ v \in \Sigma^*\ :\ \exists u \in S, w \in \underline{b}(\{v\}) \text{ s.t. } u \cdot w \in K_\Sigma\} \\ & = & \{\ v \in \Sigma^*\ :\ \exists u \in S, w \in \underline{b}(\{v\}) \text{ s.t. } rev(w) \cdot rev(u) \in rev(K_\Sigma)\} \\ & = & \{\ v \in \Sigma^*\ :\ \exists u \in rev(S), w \in rev(\underline{b}(\{v\})) \text{ s.t. } w \cdot u \in rev(K_\Sigma)\} \\ & = & \underline{b}^+(rev(rev(K_\Sigma)/rev(S))) \end{array}$$

Case 2.6 $a = b \cap K_\Sigma$ (Fact 2.7)

$$\begin{array}{rcl} \underline{a}^+(S) & = & \{\ v \in \Sigma^*\ :\ \exists u \in S \text{ s.t. } u \in (\underline{b}(\{v\}) \cap K_\Sigma)\} \\ & = & \{\ v \in \Sigma^*\ :\ \exists u \in S \cap K_\Sigma \text{ s.t. } u \in \underline{b}(\{v\})\} \\ & = & \underline{b}^+(S \cap K_\Sigma) \end{array}$$

Case 2.7 $a = hd(b)$ (Fact 2.8)

$$\begin{array}{rcl} \underline{a}^+(S) & = & \{\ v \in \Sigma^*\ :\ \exists\ u \in S\ \text{ s.t. } u \in hd(\underline{b}(v))\} \\ & = & \{\ v \in \Sigma^*\ :\ \exists\ \sigma \in S \cap \Sigma,\ q \in \Sigma^*\ \text{ s.t. }\ \sigma \cdot q \in \underline{b}(v)\} \\ & = & \underline{b}^+(((S \cap \Sigma) \cdot \Sigma^*)) \end{array}$$

i.e. if $w \in S$ but $w \notin \Sigma$ then $\underline{a}^+(\{w\}) = \emptyset$, if $\sigma \in S \cap \Sigma$, then σ attacks every argument v such that there in an element of $\underline{b}(\{v\})$ having the form $\sigma \cdot z$.

Case 2.8 $a = tl(b)$ (Fact 2.9)

$$\underline{a}^+(S) \ = \ \{\ v \in \Sigma^*\ :\ \exists u \in S \text{ s.t. } u \in tl(\underline{b}(\{v\})\}$$

from which $\underline{a}^+(S) = \underline{b}^+(\Sigma \cdot S)$. □

Proposition 8. Let $\langle \mathcal{M}, a \rangle$ be an AFS with $L(\mathcal{M}) = \mathcal{X}$ and $a \in \mathcal{AE}(\Sigma)$. Define the mapping, $\pi_a^+ : 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ by

$$\pi_a^+(S) = \{\ v \in \mathcal{X}\ :\ \exists\ u \in S \text{ s.t. } u \to_a v\}$$

It holds that $\pi_a^+(S) = \underline{a}_{\mathcal{X}}^+(S)$.

Proof. By definition $\pi_a^+(S) = \{\ v \in \mathcal{X}\ :\ \exists\ u \in S \text{ s.t. } u \in \underline{a}(v) \cap \mathcal{X}\}$, which, since $S \subseteq \mathcal{X}$, is equivalent to $\{\ v \in \mathcal{X}\ :\ \exists\ u \in S \text{ s.t. } u \in \underline{a}(v)\} = \{\ v \in \Sigma^*\ :\ \exists\ u \in S \text{ s.t. } u \in \underline{a}(v)\} \cap \mathcal{X} = \underline{a}^+(S) \cap \mathcal{X} = \underline{a}_{\mathcal{X}}^+(S)$. □

Appendix B.3. Proofs of Section 6

Theorem 3. Let $\langle \mathcal{M}, a\rangle$ be an AFS, with induced argumentation framework $\langle \mathcal{X}, \mathcal{A}\rangle$ and $S \subseteq \mathcal{X}$. The following problems are decidable.

a. Deciding if the set S is conflict free
b. For $x \in \mathcal{X}$, deciding if $x \in \mathcal{F}(S)$, i.e. whether x is acceptable to S
c. Deciding if $S \in \mathcal{E}_{adm}(\langle \mathcal{X}, \mathcal{A}\rangle)$, i.e. whether S is admissible
d. Deciding if $S \in \mathcal{E}_{stab}(\langle \mathcal{X}, \mathcal{A}\rangle)$, i.e. whether S is a stable extension
e. Constructing a DFA accepting $\mathcal{F}(S) = \mathcal{X} \setminus \pi_a^+(\mathcal{X} \setminus \pi_a^+(S))$, i.e. the set of arguments acceptable to S.
f. Deciding if $S \in \mathcal{E}_{comp}(\langle \mathcal{X}, \mathcal{A}\rangle)$, i.e. whether S is a complete extension

Proof. We first observe that validating an instance $\langle \langle \mathcal{M}, a\rangle, \mathcal{M}_S\rangle$, where $\mathcal{M}_S$ is a finite automaton accepting S, as legal simply involves checking $L(\mathcal{M}_S) \subseteq L(\mathcal{M})$, i.e. constructing an automaton $\mathcal{M}_V$ accepting $L(\mathcal{M}_S) \setminus L(\mathcal{M})$ and checking that $L(\mathcal{M}_V) = \emptyset$ (see Fact 6(a) in Appendix A).

For (a), S is conflict free if and only if $S^+ \cap S = S^- \cap S = \emptyset$. Thus given a DFA, $\mathcal{M}_S$ with $L(\mathcal{M}_S) \subseteq \mathcal{X}$ it suffices to check that $L(\mathcal{M}_S) \cap \pi_a^-(S) = \emptyset$, i.e. construct $\mathcal{M}_{cf}$ accepting $L(\mathcal{M}_S) \cap \pi_a^-(S)$ and check that $L(\mathcal{M}_{cf}) = \emptyset$.

In (b), $x \in \mathcal{F}(S)$ if and only if $(y \in \pi_a^-(\{x\})) \Rightarrow (y \in \pi_a^+(S))$ so that $x \in \mathcal{F}(S)$ if and only if $\pi_a^+(S) \supseteq \pi_a^-(\{x\})$ which can be verified by constructing suitable automata $\mathcal{M}_S^+$ for $\pi_a^+(S)$, $\mathcal{M}_x^-$ for $\pi_a^-(\{x\})$ and checking that $L(\mathcal{M}_x^-) \setminus L(\mathcal{M}_S^+) = \emptyset$.

For (c), $S \in \mathcal{E}_{adm}(\langle \mathcal{X}, \mathcal{A}\rangle)$ if and only if S is conflict free, which can be verified using the result of part (a) and $S^+ \supseteq S^-$, i.e. every attacker y of an argument in S is counterattacked by some argument z of S. It follows that to check $S \in \mathcal{E}_{adm}(\langle \mathcal{X}, \mathcal{A}\rangle)$ having verified that S is conflict free requires only checking $\pi_a^-(S) \setminus \pi_a^+(S) = \emptyset$.

Part (d) follows by checking that S is conflict free and $S \cup \pi_a^+(S) = \mathcal{X}$.

To show (e), first observe that $\mathcal{X} \setminus \pi_a^+(S)$ consists of those arguments in $\mathcal{X}$ that are *not* attacked by any argument in S. It follows that any argument that is attacked by some $y \in \mathcal{X} \setminus \pi_a^+(S)$ cannot be acceptable wrt to S since S does not contain any counterattack. The set of arguments attacked by some $y \in \mathcal{X} \setminus \pi_a^+(S)$ is just $\pi_a^+(\mathcal{X} \setminus \pi_a^+(S))$ and, hence, any argument that does not belong to this set, i.e. arguments in $\mathcal{X} \setminus \pi_a^+(\mathcal{X} \setminus \pi_a^+(S))$ are acceptable to S. If S is a regular language, then since all stages preserve regularity, i.e. $\pi_a^+(S)$, $\mathcal{X} \setminus \pi_a^+(S)$, $\pi_a^+(\mathcal{X} \setminus \pi_a^+(S))$ and $\mathcal{X} \setminus \pi_a^+(\mathcal{X} \setminus \pi_a^+(S))$ are all regular, from Thm. 2 and the fact that there are effective algorithms for constructing a DFA accepting $S_1 \setminus L_2$ (see Fact 6 in Appendix A) we can construct the required DFA.

Finally (f) is immediate from (a) and (e) and the definition of $\mathcal{E}_{comp}$. □

Theorem 4. Let $\langle \mathcal{M}, a\rangle$ be an AFS in which the induced argumentation framework $\langle \mathcal{X}, \mathcal{A}\rangle$ is finitary. The following problems are all *semi-decidable*.

a. Determining if $\mathcal{E}_{stab}(\langle \mathcal{X}, \mathcal{A}\rangle) = \emptyset$.
b. Given a *finite* $R \subset \mathcal{X}$, determining if $\forall\ w \in R\ \neg \mathrm{CA}_{adm}(\langle \mathcal{X}, \mathcal{A}\rangle, w)$.

Proof. The approach used is similar for both results and exploits the (propositional) form of the so-called *Compactness Theorem*[32].

The *lexicographic ordering*, $\leq_{\text{lex}}$ of $\Sigma^* \setminus \{\varepsilon\}$ has $u \leq_{\text{lex}} v$ if $|u| < |v|$ or (when $|u| = |v|$) if $u = \sigma_i x,\ v = \sigma_j y$ and $i < j$ or when $u = \sigma_i x,\ v = \sigma_i y$ if either $x = y = \varepsilon$ or $x \leq_{\text{lex}} y$. We use w_i to denote the i'th word in $\Sigma^* \setminus \{\varepsilon\}$ under this ordering.

Let $Z = \{z_1,\ z_2, \ldots, z_k, \ldots\}$ be an enumerable infinite set of propositional variables and define a bijective mapping $\chi\ :\ Z \leftrightarrow \Sigma^*$ via $\chi(z_i) = w_i$. For part (a) consider the following collection of clauses $\varphi(Z)$:

$$\varphi(Z) \quad = \quad CONF(Z) \bigwedge RANGE(Z)$$

where

$$CONF(Z) \quad = \quad \bigwedge_{\{z_i\ :\ \chi(z_i) \in \mathcal{X}\}} \quad \bigwedge_{\{z_j\ :\ \chi(z_j)\ \in \mathcal{X}\ \ \&\ \chi(z_j) \to_a \chi(z_i)\}} (\neg z_i \vee \neg z_j)$$

$$RANGE(Z) \quad = \quad \bigwedge_{\{z_i\ :\ \chi(z_i) \in \mathcal{X}\}} \left(z_i \quad \vee \quad \bigvee_{\{z_j\ :\ \chi(z_j)\ \in \mathcal{X}\ \ \&\ \chi(z_j) \to_a \chi(z_i)\}} z_j \right)$$

Thus if $S \in \mathcal{E}_{stab}(\langle \mathcal{X}, \mathcal{A} \rangle)$ then the assignment $z_i = \top$ iff $\chi(z_i) \in S$ will satisfy $\varphi(Z)$ and, conversely, if $\langle \alpha_1, \alpha_2, \ldots, a_k, \ldots \rangle$ is a satisfying assignment to Z for $\varphi(Z)$ then the subset $\{\chi(z_i)\ :\ \alpha_i = \top\} \in \mathcal{E}_{stab}(\langle \mathcal{X}, \mathcal{A} \rangle)$. It follows that $\mathcal{E}_{stab}(\langle \mathcal{X}, \mathcal{A} \rangle) = \emptyset$ if and only if $\varphi(Z)$ is unsatisfiable, and hence via the Compactness Theorem, if and only if there is a *finite* subset of clauses from $\varphi(Z)$ that are collectively unsatisfiable.

For any subset S of Σ^* let

$$Cl(S) \quad = \quad conf(S) \bigwedge range(S)$$

where

$$conf(S) \quad = \quad \bigwedge_{\{z_i\ :\ \chi(z_i) \in S\}} \quad \bigwedge_{\{z_j\ :\ \chi(z_j)\ \in \mathcal{X}\ \ \&\ \chi(z_j) \to_a \chi(z_i)\}} (\neg z_i \vee \neg z_j)$$

$$range(S) \quad = \quad \bigwedge_{\{z_i\ :\ \chi(z_i) \in S\}} \left(z_i \quad \vee \quad \bigvee_{\{z_j\ :\ \chi(z_j)\ \in \mathcal{X}\ \ \&\ \chi(z_j) \to_a \chi(z_i)\}} z_j \right)$$

For a finite S both $conf(S)$ and $range(S)$ are finite since for each element z_i of S the set of elements z_j corresponding to its attackers is finite. Moreover

[32]The property that $\varphi(Z)$, an infinite collection of finite clauses – or, more generally, finite propositional formulae – over an enumerable collection of propositional variables, is satisfiable if and only if every *finite* subset of clauses from $\varphi(Z)$ is satisfiable.

note that the set of clauses in $Cl(S)$ is strictly monotonic wrt inclusion (since each additional element z_i entails the addition of at least a clause in $range(S)$) and that for each clause in $CONF(Z)$ and $RANGE(Z)$ there is a z_i such that if $z_i \in S$ then the clause belongs to $Cl(S)$.

Now consider the increasing (wrt inclusion) sequence of finite subsets S of Σ^* obtained by adding incrementally the k-th element of $\Sigma^* \setminus \{\varepsilon\}$ in the lexicographic order. Then for any finite subset Q of clauses from $\varphi(Z)$ it is clearly the case that there is some S in the sequence for which $Q \subseteq Cl(S)$.

These observations yield the method given in Alg. 2.

Algorithm 2 Semi-decision process for $\mathcal{E}_{stab}(\langle \mathcal{X}, \mathcal{A} \rangle) = \emptyset$ in (finitary) AFS

```
1: S := ∅;
2: k := 0
3: while undecided do
4:    k := k + 1;
5:    S := S ∪ { w_k };
6:    if Cl(S) is unsatisfiable then
7:       report true
8:    end if
9: end while
```

To establish correctness it is sufficient to note that, by the compactness theorem, $\varphi(Z)$ is unsatisfiable iff some finite subset Q of its clauses is so, hence iff there is some finite set $S_Q \subseteq \mathcal{X}$ with $Q \subseteq Cl(S_Q)$ yielding an unsatisfiable subset of clauses. Since S_Q is finite, such a subset will eventually have $S_Q \subseteq S$ in the algorithm iff $\varphi(Z)$ is unsatisfiable, i.e. the Alg. 2 will terminate whenever $\mathcal{E}_{stab}(\langle \mathcal{X}, \mathcal{A} \rangle) = \emptyset$.

For part (b), the formula $\varphi(Z) = CONF(Z) \wedge DEF(Z) \wedge IN(R)$ is used, where $CONF(Z)$ is as previously, $DEF(Z)$ is

$$\bigwedge_{\{z_i \,:\, \chi(z_i) \in \mathcal{X}\}} \; \bigwedge_{\{z_j \,:\, \chi(z_j) \in \mathcal{X} \;\&\; \chi(z_j) \rightarrow_a \chi(z_i)\}} \left(\neg z_i \;\vee \bigvee_{\{z_k \,:\, \chi(z_k) \in \mathcal{X} \;\&\; \chi(z_k) \rightarrow_a \chi(z_j)\}} z_k \right)$$

and

$$IN(R) \;=\; \bigvee_{\{z \,:\, \chi(z) \in R\}} z$$

A similar procedure is used to that of Alg. 2, however S in l. 1 is initiated to R (the finite subset of $\mathcal{X}$ forming part of the problem instance) and $Cl(S)$ in l. 6 is replaced by $conf(S) \wedge def(S) \wedge IN(R)$, so that $\varphi(Z)$ is satisfiable iff every finite subset of its clauses that include the clause $IN(R)$ is satisfiable: note that the assignment $z_i := \bot$ for all i will satisfy every finite subset of $\varphi(Z) \setminus \{IN(R)\}$. □

Proposition 9. Let $\langle \mathcal{M}, a \rangle$ be an AFS with induced argumentation framework $\langle \mathcal{X}, \rightarrow_a \rangle$. Let $\mathcal{K} = \{K^1_\Sigma, K^2_\Sigma, \ldots, K^r_\Sigma\}$ be the set of regular expressions used

in defining a, i.e. with operations $b \cdot K_\Sigma$, $K_\Sigma \cdot b$, K_Σ/b, b/K_Σ and $b \cap K_\Sigma$. If *no* $K \in \mathcal{K}$ uses the $*$ operator then $\langle \mathcal{X}, \rightarrow_a \rangle$ is finitary.

Proof. Suppose $a \in \mathcal{AE}(\Sigma)$ satisfies the conditions of the proposition statement. Consider any $x \in \mathcal{X}$. If, in contradiction to the claim, $\pi_a^-(\{x\})$ is unbounded then $\underline{a}(\{x\})$ must yield an infinite language. Let a be a smallest (wrt size) member of $\mathcal{AE}(\Sigma)$ with this property. Clearly $size(a) > 0$ since all a with $size(a) = 0$ have $|\underline{a}(\{x\})| = 1$. Then a must have one of the forms $\{b \cup c, K_\Sigma \cdot b, b \cdot K_\Sigma, K_\Sigma/b, b/K_\Sigma, b \cap K_\Sigma, tl(b), hd(b)\}$, where $size(b) < size(a)$ and $size(c) < size(a)$, hence $|\underline{b}(\{x\})|$ and $|\underline{c}(\{x\})|$ are finite. The expression, K_Σ, uses only operations from $\{\cdot, +\}$ and it is easily shown that these cannot generate an infinite subset of Σ^*. Then it is easy to see that all the operators above give rise to a finite language, i.e. $\underline{a}(\{x\})$ is finite. □

References

[1] L. Amgoud, L. Bodenstaff, M. Caminada, P. McBurney, S. Parsons, H. Prakken, J. van Veenen, , and G. Vreeswijk. Final review and report on formal argumentation system. Technical Report Deliverable D2.6, ASPIC Project IST-FP6-002307, 2006.

[2] L. Amgoud and C. Cayrol. A reasoning model based on the production of acceptable arguments. *Annals of Math. and AI*, 34:197–215, 2002.

[3] Robert Axelrod and William D. Hamilton. The evolution of cooperation. *Science*, 211:1390–1396, 1981.

[4] P. Baroni, F. Cerutti, P.E. Dunne, and M. Giacomin. Computing with infinite argumentation frameworks: the case of AFRAs. In *Proc. 1st Intnl. Workshop on Theory and Appl. of Formal Argumentation (TAFA)*, 2011.

[5] P. Baroni, F. Cerutti, M. Giacomin, and G. Guida. AFRA: argumentation framework with recursive attacks. *Int. J. Approx. Reason.*, 51(1):19–37, 2011.

[6] P. Baroni, P.E. Dunne, and M. Giacomin. On the resolution-based family of abstract argumentation semantics and its grounded instance. *Artificial Intelligence*, 175(3-4):791–813, 2011.

[7] P. Baroni and M. Giacomin. Semantics of abstract argument systems. In I. Rahwan and G. Simari, editors, *Argumentation in AI*, chapter 2, pages 25–44. Springer-Verlag, 2009.

[8] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *Knowledge Engineering Review*, 26(4):365–410, 2011.

[9] Pietro Baroni, Massimiliano Giacomin, and Giovanni Guida. Extending abstract argumentation systems theory. *Artificial Intelligence*, 120(2):251–270, 2000.

[10] Pietro Baroni, Massimiliano Giacomin, and Giovanni Guida. Self-stabilizing defeat status computation: dealing with conflict management in multi-agent systems. *Artificial Intelligence*, 165(2):187–259, 2005.

[11] Sabrina Baselice and Piero A. Bonatti. A decidable subclass of finitary programs. *Theory and Practice of Logic Programming*, 10(4-6):481–496, 2010.

[12] Sabrina Baselice, Piero A. Bonatti, and Giovanni Criscuolo. On finitely recursive programs. In *Proc. of 23rd International Conference on Logic Programming (ICLP 2007)*, volume 4670 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 2007.

[13] D. R. Bean. Effective coloration. *Jnl. of Symbolic Logic*, 41:469–480, 1976.

[14] T. Bench-Capon and P. E. Dunne. Argumentation in artificial intelligence. *Artificial Intelligence.*, 171(10-15):619–641, 2007.

[15] T. J. M. Bench-Capon. Persuasion in Practical Argument Using Value-based Argumentation Frameworks. *Journal of Logic and Computation*, 13(3):429–448, 2003.

[16] Philippe Besnard and Anthony Hunter. *Elements of Argumentation*. MIT Press, 2008.

[17] Antonis Bikakis and Grigoris Antoniou. Defeasible contextual reasoning with arguments in ambient intelligence. *IEEE Trans. Knowledge and Data Engineering*, 22(11):1492–1506, 2010.

[18] J.-C. Birget. Intersection and union of regular languages and state complexity. *Inf. Process. Lett.*, 43:185–190, September 1992.

[19] A. Blumensath and E. Grädel. Finite presentations of infinite structures: automata and interpretations. *Theory Comput. Systems*, 37:641–674, 2004.

[20] A. Bochman. Collective argumentation and disjunctive logic programming. *Journal of Logic and Computation*, 13(3):405–428, 2003.

[21] Piero A. Bonatti. Reasoning with infinite stable models. *Artificial Intelligence*, 156(1):75–111, 2004.

[22] Piero A. Bonatti. Erratum to: Reasoning with infinite stable models [artificial intelligence 156 (1) (2004) 75-111]. *Artificial Intelligence*, 172(15):1833–1835, 2008.

[23] A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93(1–2):63–101, 1997.

[24] G. Brewka, P. E. Dunne, and S. Woltran. Relating the semantics of abstract dialectical frameworks and standard AFs. In T. Walsh, editor, *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI-2011)*, pages 780–785, 2011.

[25] G. Brewka and S. Woltran. Abstract dialectical frameworks. In *Proc. Principles of Knowledge Representation and Reasoning*, pages 102–111, 2010.

[26] Francesco Calimeri, Susanna Cozza, Giovambattista Ianni, and Nicola Leone. Computable functions in asp: Theory and implementation. In *Proc. 24th International Conference on Logic Programming (ICLP 2008)*, volume 5366 of *Lecture Notes in Computer Science*, pages 407–424. Springer, 2008.

[27] Francesco Calimeri, Susanna Cozza, Giovambattista Ianni, and Nicola Leone. An asp system with functions, lists, and sets. In *Proc. of 10th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2009)*, volume 5753 of *Lecture Notes in Computer Science*, pages 483–489. Springer, 2009.

[28] Francesco Calimeri, Susanna Cozza, Giovambattista Ianni, and Nicola Leone. Enhancing asp by functions: Decidable classes and implementation techniques. In *Proceedings of the Twenty-Fourth AAAI Conf. on Artificial Intelligence (AAAI 2010)*. AAAI Press, 2010.

[29] M. Caminada. Semi-stable semantics. In P. E. Dunne and T. J. M. Bench-Capon, editors, *Proc. 1st Int. Conf. on Computational Models of Argument*, volume 144 of *FAIA*, pages 121–130. IOS Press, 2006.

[30] M. Caminada and B. Verheij. On the existence of semi-stable extensions. In *Proc. 22nd Benelux Conference on Artificial Intelligence (BNAIC)*, 2010.

[31] T. Colcombet. On families of graphs having a decidable first order theory with reachability. In *Proc. ICALP*, volume 2380 of *LNCS*, pages 98–109. Springer-Verlag, 2002.

[32] S. Coste-Marquis, C. Devred, and P. Marquis. Constrained argumentation frameworks. In *Proc. Principles of Knowledge Representation and Reasoning*, pages 112–122, 2006.

[33] B. Courcelle. The monadic second-order logic of graphs, II: infinite graphs of bounded width. *Math. Systems Theory*, 21:187–221, 1989.

[34] Colin de la Higuera. A bibliographical study of grammatical inference. *Pattern Recognition*, 38(9):1332–1348, 2005.

[35] C. Devred, S. Doutre, C. Lefevre, and P. Nicolas. Dialectical proofs for constrained argumentation. In *Proc. 3rd COMMA*, volume 216 of *FAIA*, pages 159–170. IOS Press, 2010.

[36] Y. Dimopoulos and A. Torres. Graph theoretical structures in logic programs and default theories. *Th. Comp. Sci.*, 170:209–244, 1996.

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

[38] P. E. Dunne. *Computability Theory – concepts and applications.* Ellis–Horwood, 1991.

[39] P. E. Dunne. Prevarication in dispute protocols. In *Proc. 9th Intnl. Conf. on AI and Law (ICAIL)*, pages 12–21. ACM Press, 2003.

[40] P. E. Dunne. The Computational Complexity of Ideal Semantics. *Artificial Intelligence*, 173(18):1559–1591, 2009.

[41] P. E. Dunne and T. J. M. Bench-Capon. Coherence in finite argument systems. *Artificial Intelligence*, 141:187–203, 2002.

[42] P. E. Dunne and T. J. M. Bench-Capon. Two party immediate response disputes: properties and efficiency. *Artificial Intelligence*, 149:221–250, 2003.

[43] P. E. Dunne, A. Hunter, P. McBurney, S. Parsons, and M. Wooldridge. Weighted argument systems: Basic definitions, algorithms, and complexity results. *Artificial Intelligence*, 175(2):457–486, 2011.

[44] P. E. Dunne and M. Wooldridge. Complexity of abstract argumentation. In I. Rahwan and G. Simari, editors, *Argumentation in AI*, chapter 5, pages 85–104. Springer-Verlag, 2009.

[45] W. Dvořák, P. E. Dunne, and S. Woltran. Parametric properties of ideal semantics. In T. Walsh, editor, *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI-2011)*, pages 851–856, 2011.

[46] W. Dvořák and S. Woltran. Complexity of semi-stable and stage semantics in argumentation frameworks. *Inf. Process. Lett.*, 110:425–430, May 2010.

[47] Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran. Answer-set programming encodings for argumentation frameworks. *Argument and Computation*, 1(2):147–177, 2010.

[48] Thomas Eiter and Mantas Simkus. $\mathbb{FDNC}$: Decidable nonmonotonic disjunctive logic programs with function symbols. *ACM Trans. on Computational Logic*, 11(2), 2010.

[49] A. Fraenkel. Planar kernel and grundy with $d \leq 3$, $d_{out} \leq 2$, $d_{in} \leq 2$ are NP–complete. *Discrete Appl. Math.*, 3(4):257–262, 1981.

[50] D. Gabbay and J. Woods. More on non-cooperation in dialogue logic. *Logic Journal of IGPL*, 9(1):305–323, 2001.

[51] D. Gabbay and J. Woods. Non-cooperation in dialogue logic. *Synthese*, 127:161–186, 2001.

[52] A. J. Garcia and G. R. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(2):95–138, 2004.

[53] Nikos Gorogiannis and Anthony Hunter. Instantiating abstract argumentation with classical logic arguments: Postulates and properties. *Artificial Intelligence*, 175(9-10):1479–1497, 2011.

[54] Guido Governatori, Michael J. Maher, Grigoris Antoniou, and David Billington. Argumentation semantics for defeasible logic. *J. of Logic and Computation*, 14(5):675–702, 2004.

[55] Erich Grädel, Wolfgang Thomas, and Thomas Wilke. Preface to automata, logics, and infinite games. In *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

[56] J. E. Hopcroft. An $n \log n$ algorithm for minimizing the states in a finite automaton. In Z. Kohavi and A. Paz, editors, *The Theory of Machines and Computations*, pages 189–196. Academic Press, 1971.

[57] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[58] G. Jirásková. State complexity of some operations on binary regular languages. *Theoretical Computer Science*, 330(2):287–298, 2005. Descriptional Complexity of Formal Systems.

[59] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Log.*, 7(3):499–562, 2006.

[60] Yuliya Lierler and Vladimir Lifschitz. One more decidable class of finitely ground programs. In *Proc. 25th Int. Conf. on Logic Programming (ICLP 2009)*, volume 5649 of *Lecture Notes in Computer Science*, pages 489–493. Springer, 2009.

[61] Peter McBurney and Simon Parsons. Representing epistemic uncertainty by means of dialectical argumentation. *Annals of Mathematics and Artificial Intelligence*, 32(1-4):125–169, 2001.

[62] Peter McBurney and Simon Parsons. Dialogue games for agent argumentation. In Guillermo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 261–280. Springer US, 2009.

[63] Elliot Mendelson. *Introduction to Mathematical Logic*. CRC Press, 2010.

[64] S. Modgil. Reasoning about preferences in argumentation frameworks. *Artificial Intelligence*, 173(9–10):901–934, 2009.

[65] F. R. Moore. On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. *IEEE Trans. Comput.*, 20:1211–1214, October 1971.

[66] C. Morvan. On rational graphs. In *Proc. FOSSACS*, volume 1784 of *LNCS*, pages 252–266. Springer-Verlag, 2000.

[67] D. Muller and P. Schupp. The theory of sets, pushdown automata, and second-order logic. *Theor. Comp. Sci.*, 37:51–75, 1985.

[68] A. Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):pp. 541–544, 1958.

[69] S. H. Nielsen and S. Parsons. Computing preferred extensions for argumentation systems with sets of attacking arguments. In P. E. Dunne and T. J. M. Bench-Capon, editors, *Proc. 1st COMMA*, volume 144 of *FAIA*, pages 97–108. IOS Press, 2006.

[70] Simon Parsons, Carles Sierra, and Nicholas R. Jennings. Agents that reason and negotiate by arguing. *J. of Logic and Computation*, 8(3):261–292, 1998.

[71] G. Pighizzini and J. Shallit. Unary language operations, state complexity and jacobsthal's functions.2002. *Intl. J. of Foundations of Comp. Sci.*, 13:145–159, 2002.

[72] John L. Pollock. How to reason defeasibly. *Artificial Intelligence*, 57(1):1–42, 1992.

[73] John L. Pollock. Justification and defeat. *Artificial Intelligence*, 67(2):377–407, 1994.

[74] John L. Pollock. Perceiving and reasoning about a changing world. *Computational Intelligence*, 14(4):498–562, 1998.

[75] H. Prakken. Coherence and flexibility in dialogue games for argumentation. *J. Log. and Comput.*, 15:1009–1040, December 2005.

[76] Henry Prakken. An abstract framework for argumentation with structured arguments. *Argument and Computation*, 1(2):93–124, 2010.

[77] I. Rahwan and G. Simari, editors. *Argumentation in AI*. Springer-Verlag, 2009.

[78] Iyad Rahwan. Guest editorial: Argumentation in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 11(2):115–125, 2005.

[79] H. G. Rice. On completely recursively enumerable sets and their key arrays.3. *Jnl. Symbolic Logic*, 21:304–341, 1956.

[80] Barkley Rosser. An informal exposition of proofs of gödel's theorems and church's theorem. *The Journal of Symbolic Logic*, 4(2):pp. 53–60, 1939.

[81] Guillermo Ricardo Simari and Ronald Prescott Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence*, 53(2-3):125–157, 1992.

[82] Mantas Simkus and Thomas Eiter. $\mathbb{FDNC}$: Decidable non-monotonic disjunctive logic programs with function symbols. In *Proc. 14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning(LPAR 2007)*, volume 4790 of *Lecture Notes in Computer Science*, pages 514–530. Springer, 2007.

[83] Osvaldo Soriano. *Cuentos de los años felices*. Editorial Sudamericana, 1993.

[84] Matthew South, Gerard Vreeswijk, and John Fox. Dungine: a java dung reasoner. In Philippe Besnard, Sylvie Doutre, and Anthony Hunter, editors, *Proc. of the 2nd Int. Conf. on Computational Models of Argument(COMMA 2008)*, volume 172 of *Frontiers in Artificial Intelligence and Applications*, pages 360–368, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press.

[85] W. Thomas. The reachability problem over infinite graphs. In *Proc. CSR*, volume 5675 of *LNCS*, pages 12–18. Springer-Verlag, 2009.

[86] A. Thue. Die losung eines spezialfalles eines allgemeinen logischen problems. *Kra. Videnskabs-Selskabets Skrifter, Mat. Nat. Kl.*, 8, 1910.

[87] Bart Verheij. Deflog: on the logical interpretation of prima facie justified assumptions. *J. of Logic and Computation*, 13(3):319–346, 2003.

[88] G. Vreeswijk and H. Prakken. Credulous and sceptical argument games for preferred semantics. In *Proceedings of JELIA'2000, The 7th European Workshop on Logic for Artificial Intelligence.*, pages 224–238, Berlin, 2000. Springer LNAI 1919, Springer Verlag.

[89] G. A. W. Vreeswijk. Abstract argumentation systems. *Artificial Intelligence*, 90(1–2):225–279, 1997.

[90] Gerard Vreeswijk. Defeasible dialectics: A controversy-oriented approach towards defeasible argumentation. *J. Log. Comput.*, 3(3):317–334, 1993.

[91] Emil Weydert. Semi-stable extensions for infinite frameworks. In *Proc. of the 23nd Benelux Conference on Artificial Intelligence (BNAIC'11)*, pages 336–343, 2011.

[92] S. Yu, Q. Zhuang, and K. Salomaa. The state complexities of some basic operations on regular languages. *Theor. Comput. Sci.*, 125:315–328, March 1994.