Universidade de Évora
Instituto de Investigação e Formação Avançada

Distributed Knowledge Bases: A Proposal for Argumentation-based Semantics with Cooperation

Iara Carnevale de Almeida

orientador: *Prof. Doutor José Júlio Alferes*

co-orientador: *Prof. Doutor Luis Arriaga da Cunha*

Janeiro de 2011

Doutoramento/Ramo de Conhecimento em Informática.

Autor	Iara Carnevale de Almeida
Título	Distributed Knowledge Bases: A Proposal for Argumentation-based Semantics with Cooperation
Orientador	Prof. Doutor José Júlio Alferes
Co-Orientador	Prof. Doutor Luís Arriaga da Cunha
Instituição	Universidade de Évora Departamento de Informática
Palavras Chave	Artificial Intelligence, Extended Logic Programming, Knowledge Representation, Distributed Knowledge Bases, Paraconsistency, Incomplete knowledge, Cooperation, Negotiation Argument-based Semantics.
Endereço	Departamento de Informática Universidade de Évora Rua Romão Ramalho, 59 7000-671 Évora, Portugal
E-mail	ica@di.uevora.pt
Copyright	Universidade de Évora
Local	Évora
Data	Janeiro de 2011

Acknowledgments

In writing this dissertation, I have received the help and encouragement of many people. I would certainly fail in trying to enumerate them all. Therefore, I will only mention a few that have had a particular impact on my work, directly or indirectly.

I want to thank particularly:

- my supervisor José Júlio Alferes, for all the support given throughout the elaboration of this dissertation, for his critical and objective views during the discussions we had, and for particularly his careful and critical reading of the final version;
- my co-supervisor Luís Arriaga, for all the support given throughout the elaboration of this dissertation;
- CAPES in Brazil, for the funding of my research activity during its first years;
- the University of Évora, for all institutional support, logistic and scientific support. For all my colleagues from University of Évora, for their encouragement. Special thank you to Paulo Quaresma and Luis Rato;
- For all my colleagues at FCT/UNL, especially those from the CENTRIA research group. Special thank you to Luís Moniz Pereira and Carlos Damásio;
- Special thank you to Vasco Pedro for reviewing both English and formalism of this dissertion;
- Laura Semini and Stephan Reiff-Marganiec for their omnipresent optimism;
- Patrícia Moita, Pedro Madureira and the lovely Tiago for the reinvigorating meetings;
- Paulo and Cati Quaresma, for the great support they gave me during my stay in Portugal;

- Fernando Moura Pires, a very special colleague who I had the chance to work with, for the privilege of his friendship and for his wise advice.

Last, but not the least to:

- Teresa and Antonio, Constança, Isabel, Fernanda and Anjo, Paula, and Maria José for giving me a great support outside of the University;

- Mariana and Vasco, very special children. I feel very sorry for being absent for some important periods of their lives;

- Grandma Ermelinda, for the example of strength and perseverance (perhaps it would better to say, "stubbornness") in her life, although she is not with us any more, she will always be present in my heart;

- Grandma Juracy, for her care and attention during my infancy. I am grateful for her example of the desire to keep on living, in spite of her 93 years. I feel very sorry that our "old lady" wasn't able to wait a little longer to witness the conclusion of this extra stage in my life;

- Grandpas Eurico and Manuel, for the love they always have offered me during my infancy and my rebellious adolescence;

- my parents Rita and Paulo, for the education and guidance in the my career choices and for their constant encouragement;

- my sisters and brother, Isabela, Júlio and Jussara, for every interesting talk that we have had about choices that we have made in both personal and professional life. Quite different from each other;

- the remaining family for all the support, love and care that they have always shown towards me;

- Finally, to my daughter Helena, an adorable 2-years old child who has taught me how life can be beautiful with simple and small things.

Évora, Janeiro de 2011
Iara Carnevale de Almeida

Abstract

"Distributed Knowledge Bases: A Proposal for Argumentation-based Semantics with Cooperation"

The main objective of this dissertation is to define an argumentation-based negotiation framework for distributed knowledge bases. Knowledge bases are modelling over a multi-agent setting such that each agent possibly has an independent or overlapping knowledge base. The minimum requirement for a multi-agent setting negotiation is that agents should be able to make proposals which can then either be accepted or rejected. A higher level of sophistication occurs when recipients do not just have the choice of accepting or rejecting proposals, but have the option of making counter offers to alter aspects of the proposal which are unsatisfactory. An even more elaborate kind of negotiation is argumentation-based.

The argumentation metaphor seems to be adequate for modelling situations where different agents argue in order to determine the meaning of common beliefs. In an argumentation-based negotiation, the agents are able to send justifications or arguments along with (counter) proposals indicating why they should be accepted. An argument for an agent's belief is acceptable if the agent can argue successfully against attacking arguments from other agents. Thus, agent's beliefs are characterized by the relation between its "internal" arguments supporting its beliefs and the "external" arguments supporting the contradictory beliefs of other agents. So, in a certain sense, argumentative reasoning is based on the "external stability" of acceptable arguments in the multi-agent setting.

This dissertation proposes that agents evaluate arguments to obtain a consensus about a common knowledge by both proposing arguments or trying to build opposing arguments against them. Moreover, this proposal deals with incomplete knowledge (i.e. partial arguments) and so a cooperation process grants arguments to achieve knowledge completeness. Therefore, a negotiation of an agent's belief is seen as an argumentation-based process with cooperation; both cooperation and argumentation are seen as interlaced processes. Furthermore, each agent Ag has both set $\mathcal{A}rgue$ of argumentative agents and set $\mathcal{C}ooperate$ of cooperative agents;

every Ag must reach a consensus on its arguments with agents in $\mathcal{A}rgue$, and Ag may ask for arguments from agents in $\mathcal{C}ooperate$ to complete its partial arguments.

The argumentation-based negotiation proposal allows the modelling a hierarchy of knowledge bases representing, for instance, a business's organization or a taxonomy of some subject, and also an MAS where each agent represents "acquired knowledge" in a different period of time. Furthermore, any agent in an MAS can be queried regarding the truth value of some belief. It depends on from which agent such a belief is inferred, and also what the specification in both $\mathcal{A}rgue$ and $\mathcal{C}ooperate$ is, given the overall agents in the MAS. However, such an answer will always be consistent/paraconsistent with the agents' knowledge base involved.

This dissertation proposes a (declarative and operational) argumentation semantics for an agent's knowledge base. Furthermore, it proposes a declarative argumentation-based negotiation semantics for a multi-agent setting, which uses most of the definitions from the former semantics.

Resumo

"Bases de Conhecimento Distribuídas: Uma proposta para Semânticas baseadas em Argumentação com Cooperação"

O objectivo principal desta dissertação é definir um ambiente de negociação, baseada em argumentação, para bases de conhecimento distribuídas. As bases de conhecimentos são modeladas sobre um ambiente multi-agente tal que cada agente possui uma base de conhecimento própria. As bases de conhecimento dos diversos agentes podem ser independentes ou podem incluir conhecimentos comuns. O requisito mínimo para haver negociação num ambiente multi-agente é que os agentes tenham a capacidade de fazer propostas, que poderão ser aceites ou rejeitadas. Numa abordagem mais sofisticada, os agentes poderão responder com contra-propostas, com o intuito de alterar aspectos insatisfatórios da proposta original. Um tipo ainda mais elaborado de negociação será o *baseado em argumentação*.

A metáfora da argumentação parece ser adequada à modelação de situações em que os diferentes agentes interagem com o propósito de determinar o significado das crenças comuns. Numa negociação baseada em argumentação, as (contra-)propostas de um agente podem ser acompanhadas de argumentos a favor da sua aceitação. Um agente poderá, então, ter um argumento aceitável para uma sua crença, se conseguir argumentar com sucesso contra os argumentos, dos outros agentes, que o atacam. Assim, as crenças de um agente caracterizam-se pela relação entre os argumentos "internos" que sustentam suas crenças, e os argumentos "externos" que sustentam crenças contraditórias de outros agentes. Portanto, o raciocínio argumentativo baseia-se na "estabilidade externa" dos argumentos aceitáveis do conjunto de agentes.

Neste trabalho propõe-se uma negociação baseada em argumentação em que, para chegarem a um consenso quanto ao conhecimento comum, os agentes constroem argumentos que sustentam as suas crenças ou que se opõem aos argumentos dos agentes que as contradizem. Além disso, esta proposta lida com conhecimento incompleto (i.e. argumentos parciais) pela definição de um processo de cooperação que permite completar tal conhecimento. Assim, a negociação entre agentes é um

processo argumentativo-cooperativo, em que se podem alternar os argumentos contra e a favor das crenças de um agente. Para a formação das suas crenças, a cada agente Ag está associado um conjunto $Cooperate$ de agentes com quem coopera e um outro $Argue$ de agentes contra quem argumenta.

A negociação proposta permite a modelação de bases de conhecimento hierárquicas, representando, por exemplo, a estrutura de uma organização ou uma taxonomia nalgum domínio, e de ambientes multi-agente em que cada agente representa o conhecimento referente a um determinado período de tempo. Um agente também pode ser inquirido sobre a verdade de uma crença, dependendo a resposta do agente em questão e de quais os agentes que com ele cooperam e que a ele se opõem. Essa resposta será, no entanto, sempre consistente/paraconsistente com as bases de conhecimento dos agentes envolvidos.

Esta dissertação propõe semânticas (declarativa e operacional) da argumentação numa base de conhecimento de um agente. Partindo destas, propõe, também, semântica declarativa da negociação baseada em argumentação num ambiente multi-agente.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Negotiation is a key mechanism of interaction in a multi-agent setting. In such environments, agents often have no inherent authority over each other, and the only way they can influence the behavior of others is to persuaded them to act in particular ways. In some cases, the persuade may require little or no be convinced to effort to act in the way desired by the persuader. However, in other cases, the persuaded may be unwilling to accept the proposal initially and must be persuaded into changing its beliefs, goals, or preferences so that the proposal is accepted. In either case, the minimum requirement for negotiation is for the agents to be able to make proposals to each other which can then either be accepted or rejected. Another level of sophistication occurs when recipients do not just have the choice of accepting or rejecting proposals, but have the option of making counter offers to alter aspects of the proposal which are unsatisfactory. An even more elaborate kind of negotiation is *argumentation-based*. In argumentation-based negotiation, the agents are able to send justifications or arguments along with (counter) proposals indicating why they should be accepted. In fact,

> "While negotiation can be viewed as a process to find a solution, argumentation is needed to justify a proposed solution. Hence, it is clear that there is no negotiation without argumentation. In other words, argumentation is an integral part of negotiation" [Dun95].

The goal of argumentation-based negotiations semantics for a multi-agent setting is to deal with situations where different agents argue in order to determine the meaning of common beliefs. A belief of an agent is acceptable if the agent can argue successfully against attacking arguments from other agents. In other words, whether or not an agent believes in a proposition depends on whether or not at least one argument supporting this proposition can be successfully defended against the counter-arguments. Thus, the agent's beliefs are characterized by the

relations between its "internal" arguments supporting its beliefs and the "external" arguments supporting the contradictory beliefs of other agents. So argumentative reasoning can be viewed as based on the "external stability" of acceptable arguments in a multi-agent setting. If one views the distributed knowledge as coming from various agents in a multi-agent setting, it may happen that:

- Agents negotiate by exchanging parts of their knowledge (i.e. arguments) to obtain a consensus concerning their beliefs. In other words, in an argumentation-based negotiation, the agents evaluate arguments to obtain a consensus about a common knowledge by both proposing arguments and trying to build opposing arguments against them. Moreover, a set S of agents' knowledge bases is very often inconsistent if we consider the 'overall knowledge' of S. So, an argumentation-based negotiation should deal with contradictory arguments and also with the presence of *falsity* in S.

- An argumentation-based negotiation should deal with incomplete knowledge (i.e. partial arguments), and so a cooperation process is necessary to grant arguments to achieve knowledge completeness. Moreover, cooperation could be 'direct' between cooperative agents or 'indirect' between argumentative agents. The latter presumes that a proposed argument could be used to built a counter-argument against it.

- If we assume that every agent argues and cooperates with all agents in an argumentation-based negotiation process, the results of such a process and of the argumentation-based process (over the set of all agent's knowledge bases) should coincide. However, there are cases where these proposals do not coincide because an agent does not need to argue and/or to cooperate with all agents. This is the case, for instance, when a multi-agent setting represents a kind of hierarchy of knowledge where each agent has a partial knowledge of the overall process.

In logic programming, several ways to formalize argumentation-based semantics have been studied for a single logic program (e.g. [GDS09, arg10], and scientific events such as "Conference on Computational Models of Argument (COMMA)", "Conference on Principles of Knowledge Representation and Reasoning" (KR), "Argument, Dialog and Decision" at the International workshop on Non-Monotonic Reasoning (NMR), and the "Workshop Argumentation and Non-Monotonic Reasoning" (ArgNMR)). Intuitively, argumentation-based semantics treat the evaluation of a logic program as an argumentation process, i.e. a goal G is true if at least one argument for G cannot be successfully attacked. The ability to view logic programming as a non-monotonic knowledge representation language, in equal standing with other non-monotonic logics, brought to light the importance of defining

clear declarative semantics for logic programs, for which proof procedures (and implementations) are then defined (e.g. [Dun93, Dun95, PS97, BDKT97, Vre97, Lou98, SS02b, DMT02a, Pol01, DMT02b, GS04, Pra09]).

Note that a precise meaning (or semantics) must be associated with any logic program, in order to provide a declarative specification. *Declarative semantics* provide a mathematically precise definition of the meaning of a program, which is independent of its procedural executions, and is easy to manipulate and reason about. In contrast, an *operational semantics* is usually defined as a procedural mechanism that is capable of providing answers to queries. The correctness of such a mechanism is evaluated by comparing its behavior with the specification provided by the declarative semantics. Without the former, the user needs an intimate knowledge of the procedural aspects in order to write correct programs.

The main goal of the proposal presented in this dissertation is to define a declarative semantics for Argumentation-based Negotiation for "distributed knowledge bases", following the work in progress [dAA06, dAMA98a, dAMA98b, dAMA99, SdAMA97, dAMAS97, dAMA97]. The set of knowledge bases is viewed as a multi-agent setting (MAS) where each agent has an independent or overlapping knowledge in the MAS. Moreover, every agent Ag in an MAS argues and cooperates with a subset of the agents in the MAS, i.e. Ag has a set of argumentative agents and a set of cooperative agents. In general, little is assumed about these sets. We only impose that every agent argues and cooperates with itself because it would make little sense for an agent neither to access its own knowledge nor to obtain a consensus based upon its own knowledge. Moreover, argumentation and cooperation are viewed as "interlaced processes". The argumentation imposes the restriction that every agent should argue with other agents to evaluate its knowledge. The cooperation allows an agent to handle its incomplete knowledge with the 'help' of other agents.

The ability of associating argumentative and cooperative sets to each agent provides a flexible framework which, besides reflecting the possibly existing physical network, may serve for different purposes from the ones above. For example, for modelling knowledge over a hierarchy where each node of the hierarchy is represented by an agent that cooperates with all its inferiors, and must argue with all its superiors. Another example is modelling knowledge that evolves. Here, the "present" can use knowledge from the "past" unless this knowledge from the past is in conflict with later knowledge. This can be modelled by allowing any present node to cooperate with its past nodes, and forcing any past node to argue with future nodes. In both cases, it is important that the knowledge is not flattened, as in the union of all knowledge bases, and that the semantics is parametric on the specific Kb. I.e. it may happen that an argument is acceptable in a given (agent$_i$) Kb_i, and not acceptable in another (agent$_j$) Kb_j of the same system. Therefore,

a truth value of an agent's belief depends on which agent such a belief is inferred from, and also what the specification of both sets of cooperative and argumentative agents is, given the overall agents in the MAS.

Besides this distributed nature, the Argumentation-based Negotiation semantics also allows for paraconsistent forms of argumentation. In fact, we also have the goal to be able to deal with mutually inconsistent, and even inconsistent, knowledge bases. Moreover, when in presence of contradiction we want to obtain ways of agent reasoning, ranging from consistent (in which inconsistencies lead to no result) to paraconsistent. For achieving this, we focus on the properties of declarative semantics in what regards paraconsistency which are interesting by themselves, and independent from its distributed nature.

With this purpose, we first restrict our attention to the special case of the distributed semantics where only a single logic program is in the set of programs, i.e. we propose a semantics for an extended logic program with denials (ELPd) which represents the knowledge base of an agent. This semantics is argumentation-based, in the line of the work developed by [Dun95, PS97] for defining semantics of single extended logic programs. We propose two kind of arguments, strong argument and a weak version of the strong argument. To distinguish between them, a strong argument for a literal L will be denoted by A^s_L and its weak version by A^w_L. The weak version A^w_L is built by adding default literals to the rules of A^s_L, thus making the rules weaker (more susceptible to being contradicted/attacked). Intuitively, if there is a potential inconsistency then the weak argument is attacked, whereas the strong one is not. As such, the semantics succeeds in detecting conflicts in a paraconsistent ELPd, i.e. it deals with contradictory arguments. We also improve the notion of [PS97]'s status of an argument, and so an argument of an agent Ag is deduced as justified, overruled or defensible with respect to the Ag's set of arguments. Since the semantics deals with paraconsistency, if there exist contradictory arguments in a set of justified arguments S, a justified argument can in turn be contradictory, based on contradiction or non-contradictory w.r.t. S.

1.1 Main Contributions of this thesis

- We define a declarative semantics for Argumentation-based Negotiation for a multi-agent setting (MAS). Every agent Ag in a MAS argues and cooperates with a subset of agents in the MAS, i.e. Ag has a set of argumentative agents and a set of cooperative agents. The semantics for Argumentation-based Negotiation is composed by argumentation and cooperation. The former imposes the restriction that every agent should argue with other agents to evaluate its knowledge. The latter allows an agent to handle its incomplete knowledge with the 'help' of other cooperative agents.

- We extend [PS97]'s argumentation-based semantics for extended logic programs to deal with denials. We further propose two kind of arguments, strong arguments and weak arguments. Then, the declarative semantics for (Self-) argumentation succeeds in detecting conflicts in a paraconsistent extended logic program with denials, i.e. it deals with contradictory arguments.

- We improve the notion of [PS97]'s status of an argument, so that an argument of an agent Ag is justified, overruled or defensible with respect to the Ag's set of arguments. Since our argumentation proposal deals with paraconsistency, if there exist contradictory arguments in a set of justified arguments S, a justified argument can in turn be contradictory, based on contradiction or non-contradictory w.r.t. S.

- The truth value of an agent's belief may not always be the same depending on which kind of interaction between (strong and weak) arguments is chosen. According to the choice of interaction, we may obtain different well-founded semantics, viz. $WFSX_p$ semantics [ADP95], Grounded extension [Dun95], $WFSX$ [PA92], or WFS semantics [Prz90]. Since our argumentation is parameterized by the kind of interaction between arguments, we obtain results from a consistent way of reasoning to a paraconsistent way of reasoning.

- We develop a proof procedure for both declarative and operational (Self-) argumentation semantics.

1.2 Thesis Structure

Besides the present chapter, this dissertation comprises the following parts:

- Chapter 2 presents background material on the usage of defeasible argumentation for logic programming. This background is essential to understand our argumentation-based semantics with cooperation. We briefly present the Extended Logic Programming with denials language (denoted by ELPd), since it is the representation language for modelling the knowledge bases that we use in the remainder of the dissertation, and recall the work of [Dun95, PS97]'s argumentation semantics as a basis for attributing a meaning to the language. Since their work follows a fixpoint approach [Pol87], we first present the definitions of such an approach applied to argumentation. Then, both argumentation semantics for logic programming are presented.

- Chapter 3 presents an argumentation semantics which involves a "single" extended logic program, named *self-argumentation* semantics. We focus on the properties of a declarative semantics in what regards paraconsistency

which are interesting by themselves, and independent from its distributed nature. With this purpose, we restrict our attention to the special case of the distributed semantics, where only a "single" extended logic program with denials (ELPd) is in the set of programs. The *self-argumentation* semantics is inspired by two well known argumentation semantics, viz. [Dun95] and [PS97]. We redefine [PS97]'s definition of argument and other [PS97]'s definitions are simplified, the goal being to obtain a semantics for extended logic program with denials which represents the knowledge base of an agent. Furthermore, we propose a parameterized characteristic function so that our argumentation semantics obtains different levels of acceptability of an argument. With such a differentiation, we go through the properties of both conflict-free and contradictory sets of acceptable arguments. Therefore, we obtain both paraconsistent and consistent ways of reasoning. According to [PS97]'s definition of the status of an argument, the argument may be justified, overruled or defeasible. On top of that, we propose that a justified argument can be contradictory, based on contradiction, or non-contradictory. We then present a definition of the truth value of a conclusion G such that G is true (and contradictory, based-on-contradiction, or non-contradictory), false or undefined. Finally, we present a proof procedure for such a declarative semantics.

- Chapter 4 presents the main contribution of the dissertation: an argumentation-based negotiation semantics for distributed knowledge bases represented as extended logic programs. Such a semantics extends the argumentation semantics presented in the previous chapter by considering sets of (distributed) logic programs, rather than single ones. For specifying the ways in which the various logic programs may combine their knowledge we make use of concepts that have been developed in the areas of defeasible reasoning and multi-agent settings. In particular, we associate to each program P a cooperation set (the set of programs that can be used to complete the knowledge in P) and an argumentation set (the set of programs with which P has to reach a consensus). In this chapter, we first define a declarative semantics for argumentation-based negotiation. Then, some illustrative examples are presented. Finally, we present a general architecture for implementing the semantics.

- Chapter 5 compares related work in the areas of Defeasible Reasoning and Argumentation-based Negotiation.

- Finally, Chapter 6 goes back to the objectives drawn in the introduction, synthesizing the way how the work which unfolded throughout this dissertation has fulfilled them. Then, it outlines some future research aspects that

emerged from the work presented herein.

Chapter 2

Background on Defeasible Argumentation

This chapter presents background material on the usage of defeasible argumentation for logic programming. This background is essential to understand our argumentation-based semantics with cooperation. We briefly present the Extended Logic Programming with denials language (denoted by ELPd), since it is the representation language for modelling the knowledge bases that we use in the remainder of the dissertation, and recall the work of [Dun95, PS97]'s argumentation semantics as a basis for atributting a meaning to the language. Since their work follows a fixpoint approach [Pol87], we first present the definitions of such an approach applied to argumentation. Then, both argumentation semantics for logic programming are presented.

> "When a rule supporting a conclusion may be defeated by new information, it is said that such reasoning is defeasible. When we chain defeasible reasons to reach a conclusion, we have arguments, instead of proofs. It makes sense to require defeasible reasons for argumentation. Arguments may compete, rebutting each other, so a process of argumentation is a natural result of the search for arguments. Adjunction of competing arguments must be performed, comparing arguments in order to determine what beliefs are justified. Since we arrive at conclusions by building defeasible arguments, and since mathematical argumentation has so often called itself argumentation, we sometimes call this kind of reasoning *defeasible argumentation*."[CML00]

The field of defeasible argumentation is relatively new [1] and researchers disagree on many issues, while the formal meta-theory is still in its early stages.

A much-discussed issue is whether logics for non-monotonic reasoning should have a model-theoretic semantics or not. Traditionally, model theory has been used in logic to define the meaning of logical languages. Formulas of such languages were regarded as telling us something about reality (however defined). Model-theoretic semantics defines the meaning of logical symbols by defining what the worlds looks like if an expression with these symbols is true, and it defines logical consequence, entailment, by looking at what else must be true if the premises are true. For defaults, this means that their semantics should be in terms of what the world normally, or typically, looks like when defaults are true. Logical consequence should, in this approach, be determined by looking at the most normal worlds, models or situations that satisfy the premises.

However, [Pol91, Vre93, Lou98] have argued that the meaning of defaults should not be found in a correspondence with the reality, but in their role in dialectical inquiry. Then "a relation between premises and conclusions is defeasible" means that a certain burden of proof is induced. In this approach, the central notions of defeasible reasoning are notions like attack, rebuttal, and defeat among arguments, and these notions are not 'propositional', for which reason their meaning is not naturally captured in terms of correspondence between a proposition and the world. This approach, instead, defines 'argumentation-theoretic' semantics for such notions. The basic idea of such a semantics is to capture sets of arguments that are as large as possible, and adequately defend themselves against attacks on their members. [PV02] states that systems for defeasible argumentation contain the following five elements (although sometimes implicitly): an underlying logical language $\mathcal{L}$, definitions of how to build arguments over $\mathcal{L}$, of conflicts between arguments, and of defeat among arguments and, finally, a definition of the status of arguments which can be used to define a notion of defeasible consequence. The notions of underlying logic and argument still fit with the standard picture of what a logic system is. The remaining three elements are what makes an argumentation system a framework for defeasible argumentation.

Argumentation systems are defined on top of an underlying logical language and an associated notion of logical consequence, defining the way an argument is built. The idea is that this consequence notion is monotonic: new premises cannot invalidate arguments as arguments, but only give rise to counter-arguments. Some argumentation systems assume a particular logic (e.g. Fuzzy logic [SS02a] and Extended Logic Programming [PS97, SdAMA97, dAMAS97, dAMA98a, dAMA98b]),

[1] The argumentation-based approach which was the first logical formalization of defeasible argumentation was initiated by the philosopher John Pollock, see [Pol87, Pol92]. Pollock's proposal was initially applied to the philosophy of knowledge and justification (epistemology) [Pol74]. The first artificial intelligence paper on argumentation systems was proposed in [Lou87].

while other systems leave the underlying logic partly (e.g. [BDKT97][2] and [Pol95][3]) or wholly unspecified (e.g. [Dun95]). These later systems can be instantiated with various alternative logics, which became frameworks rather than systems.

The notion of an argument corresponds to a proof in the underlying logic language. As for the layout of arguments, in the literature of argumentation systems, three familiar basic formats can be distinguished. Sometimes arguments are defined as a tree of inferences grounded in the premises (e.g. [Nut94, Vre97]), and sometimes as a sequence of such inferences (e.g. [PS97, dAMA98b, SS02a]), i.e. as a deduction. Some systems simply define an argument as a premises–conclusion pair [BDKT97], leaving implicit that the underlying logic validates a proof of the conclusion from the premises. The argumentation system proposed by [Dun95] leaves the internal structure of an argument completely unspecified. Dung treats the notion of an argument as a primitive, and exclusively focuses on the ways arguments interact. Thus, Dung's framework is the most abstract.

In the literature, the notion of a conflict between arguments (the terms "attack" and "counter-argument" are also used) is discussed referring to three types. The first type is when arguments have contradictory conclusions, as in the well known example of Tweety: "Tweety flies because it is a bird" and "Tweety does not fly because it is a penguin". Clearly, this form of attack, which is often called *rebutting* an argument, is symmetric. The other two types of conflict are not symmetric. One is where one argument makes a non-provability assumption (as in default logic) and another argument proves what was assumed unprovable by the first. For example, an argument "Tweety flies because it is a bird, and it is not provable Tweety is a penguin", is attacked by any argument with the conclusion "Tweety is a penguin". This kind of attack is called *assumption attack*. The third type of conflict (proposed by [Pol87]) is when one argument challenges not a proposition, but a rule of inference of another argument. After Pollock, this is usually called *undercutting an inference*. Moreover, such type of conflict occurs only if the rule of inference is not deductive. To consider an example, the argument "raven_{101} is black since the observed ravens raven_1, ..., raven_{100} were black" is undercut by an argument "I saw raven_{102} which was white".

Furthermore, all these kinds of attack have a direct and an indirect version: an indirect attack is directed against a 'sub-conclusion' or a 'sub-step' of an argument A (also known as *sub-argument of* A). The notion of conflicting or attacking arguments does not embody any form of evaluation; evaluating conflicting pairs of arguments or, in other words, determining whether an attack is successful, is another element of argumentation systems. It has the form of a binary re-

[2]They propose to reformulate existing non-monotonic logics in their general framework, for instance, in applications of preferential entailment or default logic.

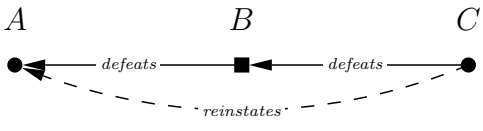
[3]The underlying logic is standart first-order logic.

lation between arguments, standing for 'attacking and not weaker' (in a weaker form) or 'attacking and stronger' (in a strong form). The terminologies vary. Some terms that have been used are: 'defeat' [Nut94, Pol95, PS97, SS02a], 'attack' [Dun95, dAMA98b, BDKT97], and 'inference' [SL92a, Lou98]. Moreover, [PS97] uses 'defeat' for the weak notion and 'strict defeat' for the strong, asymmetric notion; [Dun95] uses 'reduction ad-absurdum attack' and 'ground attack', respectively. Other systems do not explicitly name this notion of conflict but leave it implicit in the [BDKT97]'s definitions. Unless indicated otherwise, we will use the term 'defeat' in this section.

The several forms of attack, rebutting vs. assumption vs. undercutting, and direct vs. indirect have their counterparts for defeat. The notion of defeat is a binary relation on a set S of arguments. It is important to note that this relation does not yet tell us which arguments are acceptable with respect to S; it only tells us something about the relative strength of two individual conflicting arguments.

The ultimate status of an argument depends on the interaction between all arguments of S. In the following, three examples from the literature of semantics of argumentation systems viz. "Reinstatement", "Even Cycle" and "Self Defeating" are presented. These examples illustrate typical cases under which conditions of acceptability of an argument should be defined. For the moment, we do not specify the structure of an argument nor the precise definition of defeat[4]. Assume, as background, a set of arguments with a binary relation of defeat defined over it such that "A defeats B" means "A conflicts with B and A is not weaker than B". Moreover, in some cases it may happen that A defeats B and B defeats A. Assume that arguments are either 'acceptable' or 'not acceptable': an argument is acceptable if all arguments defeating it (if any) are not acceptable; otherwise, such an argument is not acceptable.

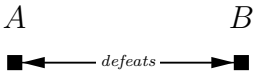
Example 1 (Reinstatement) *Consider three arguments A, B and C such that B defeats A and C defeats B. C is acceptable since it is not defeated by any other argument. This makes B not acceptable, since B is defeated by C. This in turn makes A acceptable: although A is defeated by B, A is reinstated by C. The figure below illustrates the above description; a round node represents an acceptable argument and a square node represents a not acceptable argument.*



[4]For simplicity we use the terminology of [PS97]'s proposal. As remarked above, [Dun95] uses 'attack' instead 'defeat'.

The key observation is that an argument that is defeated by another argument can only be acceptable if it is reinstated by a third argument, i.e by an acceptable argument that defeats its defeater. In case of 'undecided conflicts', a situation may be circular or ambiguous. It is not clear which argument should remain undefeated, especially when arguments of equal strength interfere with each other.

Example 2 (Even Cycle) *Consider the arguments A and B such that A defeats B and B defeats A. Intuitively, A is acceptable if B is not acceptable. However, B can also be acceptable if A is not acceptable. Thus, both cannot be acceptable at the same time.*



Finally, there is the problem of self-defeating arguments, i.e. arguments that defeat themselves.

Example 3 (Self Defeating) *Consider an argument A such that A defeats itself. If we assume that A is not acceptable, then all arguments defeating A are not acceptable, and thus it should be acceptable. This is a contradiction. If we assume that A is acceptable, then A is defeated by an acceptable argument. This is another contradiction.*

What is also needed is a definition of the status of arguments on the basis of all the ways in which they interact. Besides reinstatement, this definition must also capture the 'compositional principle' [Vre97], in which an argument cannot be acceptable unless all its sub-arguments are acceptable. There is a close relationship between these two notions, since reinstatement often proceeds by indirect attack, i.e. attacking a sub-argument of the attacking argument. The definition of the status of arguments by [Dun95] produces the output of an argumentation system, which typically divides arguments in to at least two classes: acceptable arguments, and arguments defeated by at least one acceptable argument. Sometimes a third intermediate category is also distinguished, e.g. the arguments that leave the acceptability undecided [PS97]. The terminology varies here also: terms that have been used are justified vs. defensible vs. defeated (or overruled) [PS97, dAMA98b,

SS02a], defeated vs. undefeated [Pra93, Pol95, Vre97, Lou98], preferred vs. not preferred [BDKT97, PS97], etc.

Furthermore, the status of arguments might be obtained by either a consistent or a paraconsistent way of reasoning. When faced with an unresolvable conflict between two arguments, a 'consistent reasoner' would refrain from drawing any conclusion, while a 'paraconsistent reasoner' would choose one conclusion at random (or both alternatively) and further explore its consequences. The consistent approach is often defended by saying that, since in an unresolvable conflict, no argument is stronger than another, neither of them can be justified; the paraconsistent approach has sometimes been defended by saying that the practical circumstances often require a decision about which conclusion is the best for the moment. Thus, a paraconsistent reasoner might deal with contradictory conclusions through an argumentation process. For instance, consider the arguments from the well-known "Nixon Diamond" problem: "Nixon was a pacifist because he was a quaker" and "Nixon was not a pacifist because he was a republican". A consistent reasoner neither concludes that "Nixon was a pacifist" nor that "Nixon was not a pacifist"; a paraconsistent reasoner may choose one of them at random.

The general features of argumentation-based systems can be organized along two main approaches: unique-status-assignment and multiple-status-assignment. The *unique-status-assignment* basically comes in two variants. The first variant defines status assignment in terms of a fixpoint operator which for each set of arguments returns the set of all arguments that are acceptable to it, e.g. [Pol87, Pol92, SL92a, Vre97, Dun95, dAMA98b]. The second variant involves an explicitly recursive definition of justified arguments, reflecting the basic intuition that an argument cannot be justified if not all its sub-arguments are justified, e.g. [Nut94, Pra93]. The *multiple-status-assignment* deals with competing arguments of equal strength by letting them induce two alternative status assignments, in both of which one is justified at the expense of the other(e.g. [Dun95, Pol95]); in this approach, an argument is 'genuinely' justified iff it receives this status in all status assignments. A full discussion of these approaches is beyond the scope of this work — see details in e.g. [PV02].

We are in line with the proposals of [Dun95] and [PS97]. Since both work with the fixpoint approach, we have paid special attention to it. In a declarative form with fixpoint definitions, certain sets of arguments are just declared as acceptable (given a set of premises and evaluation criteria) without defining a procedure for testing whether an argument is a member of this set. The procedural form amounts to defining such a procedure. Thus, the declarative form of an argumentation system can be regarded as its (argumentation-theoretic) semantics, and the procedural form as its proof theory.

In the remainder of this chapter, we briefly present *Extended Logic Programming* (ELP) and so an extension of ELP, viz. *Extended Logic Programming with denials* (ELPd). The ELPd is the representation language for modelling the knowledge bases that we use in the remainder of this dissertation. Then, we present the definitions of fixpoint approach applied to argumentation. Finally, both [Dun95]'s and [PS97]'s argumentation semantics for logic programming are presented.

2.1 Extended Logic Programming with Denials

Due to its declarative nature, as well as its procedural implementations, logic programming is a good language for knowledge representation. In fact, much work has been devoted to the use of logic programs for knowledge representation[5], and their relation to other well-known non-monotonic formalisms for knowledge representation and defeasible reasoning, such as default logics [Rei80] and auto-epistemic logics [Moo85]. Default logics draw plausible inferences in the absence of information, it is like arguing with Nature where a conclusion supported by argument can be drawn in the absence of any counterargument. Auto-epistemic logics reasoning about one's own knowledge or beliefs, which is much like arguing with oneself.

Normal logic programs use a non-monotonic form of "default negation"[6] whose major distinction from the classical negation is that it can be assumed in the absence of evidence to the contrary. Default negated literals are viewed as hypotheses which, under certain conditions, can be assumed. For instance, we can express default-statements of the form

Normally, unless something abnormal holds, then A implies B

A typical example for such a statement is "Birds, not shown to be abnormal, fly" and it can be represented by the following rule:

$$fly(X) \leftarrow bird(X), not\ \ abnormal(X)$$

We can further represent "Let's go swimming if it is not known to be raining and the water is not known to be cold" as follows

$$swimming \leftarrow not\ \ raining, not\ \ coldWater$$

[5]See either proceedings of the "International Conference on Principles of Knowledge Representation and Reasoning" (KRR) (in *http://www.kr.org/*) or "International Conference on Logic Programming and Non-monotonic Reasoning"(LPNMR).

[6]Or "negation as failure" as it is also usually called in the literature of logic programming.

Although default negation is quite useful in various domains and application frameworks, it is not the only type of negation that is required in non-monotonic formalisms. Indeed, while default negation $not\ p$ of an atom p is always assumed 'by default', we often need to be more careful before jumping to negative conclusions. For example, it would make little sense to say $guilty \leftarrow not\ innocent$ to express the fact that being guilty is the opposite of being innocent, because it would imply that people could be considered guilty 'by default'. Moreover, in normal logic programs, the negative information is implicit, i.e. it is not possible to explicitly state falsity; propositions are assumed false if there is no reason to believe they are true. Though this is what is required in some cases, having this single form of (implicit) negation is a serious limitation in other cases. In fact, in various situations one may want to explicitly declare that something is false. In the example of being guilty we can say that "Everyone is not guilty by default". Furthermore, "Someone is guilty if there is evidence for it (e.g. the crime has been witnessed)". There is no way of doing this with normal logic programs. This is the main reason for the generalization of language of logic programs to include an explicit form of negation. An extended logic program, which is introduced by [GL90], distinguishes the two types of negation, viz. default and explicit, and enable us to deal with negation as well as default negation in program. The generalized language used is called *Extended Logic Programming* [PA92]. In Extended Logic Programming, we can express the above example by

$$\neg guilty \leftarrow not\ guilty.$$
$$guilty \leftarrow seenDoing.$$

The above rules illustrate that explicit negation is useful to represent negative information, in such a case by having a rule with explicit negation at its head. However, explicit negation may also be needed in the body of a rule. To illustrate more about the use of extended logic programming we will now present some examples extracted from [GL90, AP96].

Consider the statement "A school bus may cross railway tracks under the condition that there is no approaching train". It would be wrong to express this statement by the rule $cross \leftarrow not\ train$. The problem is that this rule allows the bus to cross the tracks when there is no information about either the presence or the absence of a train. The situation is different if explicit negation is used, i.e.

$$cross \leftarrow \neg train.$$

Then the bus is only allowed to cross the tracks if the bus driver is sure that there is no approaching train.

The difference between $not\ L$ and $\neg L$ in a logic program is essential whenever we cannot assume that available positive information about L is complete, i.e. we

cannot assume that the absence of information about L clearly denotes its falsity. Moreover, the use of explicit negation in combination with the existing default negation allows for greater expressivity. As an illustration of this improvement, consider the following rule for representing statements like "If the driver is not sure that a train is not approaching, then he should wait"; in a natural way it is depicted as

$$wait \leftarrow not\ \neg train.$$

Furthermore, general conflicts can be caused by non-complementary information, i.e. neither L and $not\ L$ nor L and $\neg L$. Consider the following statements "Let's go hiking if it is not known to be raining. Let's go swimming if it is not known to be raining and the water is not known to be cold. We cannot go both swimming and hiking". These statements can be represented as follows:

$$hiking \leftarrow not\ raining.$$
$$swimming \leftarrow not\ raining, not\ coldWater.$$
$$\bot \leftarrow hiking, swimming.$$

such that the symbol $\bot$ denotes *falsity*. In such a case, if it is neither raining nor the water is cold, it might possible to do both activities, i.e to swim and to hike. Nevertheless, it causes a conflict because both cannot be done at the same time. Thus, if both *hiking* and *swimming* hold, then *falsity* follows. We choose to represent this kind of contradiction by using the notion of integrity constraints. The basic idea on integrity constraints is that only some program states are considered acceptable, and the constraints are meant to enforce these acceptable states. Integrity constraints can be of two types, viz. static and dynamic. In static constraints, the enforcement of these constraint depends only on the current state of the program, independently of any prior state. The example above of hiking/swimming is one such an example. In dynamic constraints, these depend on two or more program states. One example is "employee's salaries can never decrease". Since it is not a purpose of this work to deal with the evolution of a program in time, dynamic integrity constraints are not addressed. We only want to deal with the problem of conflicts caused by differences between conclusions. Therefore, it is enough consider only static constraints in the form of denials.

In the remainder of this section we present the language's definition for normal logic program in order to present both definitions of Extended Logic Program and Extended Logic Program with denials.

Definition 1 (Language) *An alphabet $\mathcal{B}$ of a language $\mathcal{L}$ is a finite disjoint set of constants and predicate symbols. Moreover, the symbol $\bot \notin \mathcal{B}$. An* atom *over $\mathcal{B}$ is an expression of the form $p(t_1, \ldots, t_n)$ where p is a predicate symbol of $\mathcal{B}$ and the t_i's are terms. A* term *over $\mathcal{B}$ is either a variable or a constant. An* objective

literal *over $\mathcal{B}$ is either an atom A or its explicit negation $\neg A$. A* default literal *over $\mathcal{B}$ is of the form not A where A is an objective literal. A* literal *is either an objective literal or a default literal. By not $\{L_1, \ldots, L_n\}$ we mean the set of default literals $\{not\ L_1, \ldots, not\ L_n\}$. By (negative) hypothesis of an objective literal L we mean not L. By explicit complement of an objective literal L we mean $\neg L$ if L is an atom, or A if $L = \neg A$. A term (resp. atom, literal) is called* ground *if it does not contain variables. By the Extended Herbrand Base $\mathcal{H}$ of $\mathcal{B}$, $\mathcal{H}(\mathcal{B})$, we mean the set of all ground objective literals of $\mathcal{B}$.*

The purpose of logic programming semantics is to determine from a program P the set of literals which should hold or not, according to some logical, intuitive, or commonsensical principles. These sets of literals form interpretations. An interpretation expresses the intended meaning of the program P. For the above language, several declarative semantics have been defined, e.g. the answer-sets semantics [GL90] (which is a generalization of the stable models semantics of normal logic programs), the well-founded semantics with explicit negation ($WFSX$) [PA92], and the well-founded semantics with "classical" negation [Prz90]. $WFSX$, unlike WFS with classical negation, considers the so-called coherence requirement relating the two form of negation: "*if L is explicitly false then L must be assumed false by default*". In other words, in extended logic programs, default literals can be viewed as hypotheses, where an objective literal L inhibits the hypothesis *not* L and $\neg L$ makes the assumption of hypothesis *not* L imperative.

A paraconsistent extension of $WFSX$ ($WFSX_p$) has been defined in [ADP95] for Extended Logic Programs, presented below. In $WFSX_p$, unlike in the others mentioned above, contradictory information is accepted and dealt with by the semantics. The main idea of $WFSX_p$ is to obtain, always in keeping with coherence, all consequences of the program, even those leading to contradiction, as well as those arising from contradiction. Then, an $WFSX_p$ interpretation of an extended logic program P is a set of literals of the form $T \cup not\ F$, where T and F are subsets of $\mathcal{H}(P)$, and such an interpretation is coherent iff for every L in T we have $\neg L$ in F. Moreover, objective literals and their explicit negation are viewed as independent identities, except for the fundamental notion of coherent interpretation. This allows a pair of default contradictory literals L and $\neg L$ to belong simultaneously to T. In this case it is said that such an interpretation is contradictory or inconsistent.

Definition 2 (Extended Logic Program) *An* extended logic program (ELP) over a language $\mathcal{L}$ *is a (finite) set of (ground) rules of the form*

$$L_0 \leftarrow L_1, \ldots, L_l, not\ L_{l+1}, \ldots, not\ L_n\ (0 \leq l \leq n)$$

where each L_i *(*$0 \leq i \leq n$*) is an objective literal of* $\mathcal{L}$*. A rule is ground if all literals are ground. As usual* L_0 *is called the* head*, and* $L_1, \ldots, not\ \ L_n$ *the* body *of the rule. If* $n = 0$ *the rule is called a* fact *and the arrow symbol is omitted*

The usual reading of an extended logic program rule is that, whenever the body is true, then the head must be true. The comma "," in the body of the rule has a conjunctive flavor. For simplicity, we use non-grounded rules in the remainder. Variables are denoted with letters X, Y and Z, and constants with any other letter or with words. These rules simply stand for the ground version, i.e. ground rules are obtained by substituting in all possible ways each of the variables by elements of the Herbrand Universe.

Definition 3 (Extended Logic Program with Denials) *Let* $\mathcal{L}$ *be a language over an alphabet* $\mathcal{B}$*. A* denial *(or integrity rule) is a rule of the form*

$$\bot \leftarrow L_1, \ldots, L_l, not\ \ L_{l+1}, \ldots, not\ \ L_n\ (0 \leq l \leq n)$$

where each L_i *(*$1 \leq i \leq n$*) is an objective literal of* $\mathcal{L}$*, and the symbol* $\bot$ *stands for* falsity*. An* extended logic program with denials *(ELPd) over* $\mathcal{L}$ *is a (finite) set of (ground) rules of the form* $L \leftarrow Body$ *such that* L *is either an objective literal or the symbol* $\bot$*, and* $Body$ *is a finite set of literals.*

Let P *be an ELPd over* $\mathcal{L}$*. The* Extended Herbrand Base $\mathcal{H}$ of P *is*

$$\mathcal{H}(P) = \mathcal{H}(\mathcal{B}) \cup \{\bot\}$$

Because more adequate for our purposes, here we present $WFSX$ and $WFSX_p$in a distinctly different manner with respect to its original definition. This presentation is based on alternating fixpoints of Gelfond–Lifschitz Γ–like operators [GL90], and follows the presentation of [ADP95]. We begin by recalling the definition of the Γ operator:

Definition 4 (The Γ–operator) *Let* P *be an extended program,* I *an interpretation, and let* P' *(resp.* I'*) be obtained from* P *(resp.* I*) by denoting every literal* $\neg A$ *by a new atom, say* $\neg\_A$*. The GL–transformation* $\frac{P'}{I'}$ *is the program obtained from* P' *by removing all rules containing a default literal not* A *such that* $A \in I'$*, and by then removing all the remaining default literals from* P*. Let* J *be the least model of* $\frac{P'}{I'}$*.* ΓI *is obtained from* J *by replacing the introduced atoms* $\neg\_A$ *by* $\neg A$*.*

To impose the coherence requirement [ADP95] introduces:

Definition 5 (Semi-normal version of a program) *The semi-normal version of a program* P *is the program* P_s *obtained from* P *by adding to the (possibly empty) Body of each rule* $LlBody$ *the default literal not* $\neg L$*, where* $\neg L$ *is the complement of* L *wrt. explicit negation.*

Below we use $\Gamma(S)$ to denote $\Gamma_P(S)$, and $\Gamma_s(S)$ to denote $\Gamma_{P_s}(S)$.

Definition 6 (Partial stable model) *A set of objective literals T generates a partial stable model (PSM) of an extended program P iff:*

1. $T = \Gamma\Gamma_s T$; *and*
2. $T \subseteq \Gamma_s T$.

The partial stable model generated by T is the interpretation

$$T \cup not\ \ (\mathcal{H}(P) - \Gamma_s T)$$

Programs without partial stable models are said *contradictory*. It turns out that non-contradictory programs always have a least PSM. The *WFSX* semantics is determined by that least PSM:

Theorem 1 (*WFSX* semantics) *Every non-contradictory program P has a least (wrt. $\subseteq$) partial stable model, the well-founded model of P ($WFM(P)$).*

To obtain an iterative "bottom-up" definition for $WFM(P)$ we define the following transfinite sequence $\{I_\alpha\}$:

$$\begin{array}{rcl} I_0 & = & \{\} \\ I_{\alpha+1} & = & \Gamma\Gamma_s I_\alpha \\ I_\delta & = & \bigcup\{I_\alpha \mid \alpha < \delta\} \quad \textit{for limit ordinal } \delta \end{array}$$

There exists a smallest ordinal λ for the sequence above, such that I_λ is the smallest fixpoint of $\Gamma\Gamma_s$, and

$$WFM(P) = I_\lambda \cup not\ \ (\mathcal{H}(P) - \Gamma_s I_\lambda)$$

$WFSX_p$ generalises the *WFSX* semantic for the case of contradictory programs. *WFSX* is not defined for contradictory programs because such programs have no PSMs. By definition of PSM, a program has none if either it has no fixpoints of $\Gamma\Gamma_s$ or if all fixpoints T of $\Gamma\Gamma_s$ do not comply with $T \subseteq \Gamma_s T$. The next theorem shows that the first case is impossible, i.e. all programs (contradictory or otherwise) have fixpoints of $\Gamma\Gamma_s$.

Theorem 2 *The operator $\Gamma\Gamma_s$ is monotonic, for arbitrary sets of literals.*

Consequently every program has a least fixpoint of $\Gamma\Gamma_s$. If, for some program P, the least fixpoint of $\Gamma\Gamma_s$ complies with condition (2) of definition 6 then the program is non-contradictory and the least fixpoint is the WFM. Otherwise the program is contradictory. Moreover, the test for knowing whether a program is contradictory, given its least fixpoint of $\Gamma\Gamma_s$, can be simplified to: "the program is non-contradictory iff its least fixpoint of $\Gamma\Gamma_s$ has no pair of $\neg$-complementary literals".

Theorem 3 *Let T be the least fixpoint of $\Gamma\Gamma_s$ for a program P. Then:*

$$T \not\subseteq \Gamma_s T \quad iff \quad \exists L \in \mathcal{H},\ \{L, \neg L\} \subseteq T$$

So, if one is interested only in the WFM, the condition $T \subseteq \Gamma_s T$ can be replaced by testing whether T has $\neg$-complementary literals. Note that, in fact this condition guarantees that literals cannot be both true and false by default. By removing the condition this guarantee is no longer valid. But this is precisely what is wanted in the definition of the paraconsistent *WFSX*.

Accordingly, the definition of $WFSX_p$ is one where the construction of the WFM given by the least fixpoint of $\Gamma\Gamma_s$ is kept, condition $T \subseteq \Gamma_s T$ is removed, and where contradictory programs are those that contain a pair of complementary literals in the WFM:

Definition 7 (Paraconsistent *WFSX*) *Let P be an extended program whose least fixpoint of $\Gamma\Gamma_s$ is T. Then, the paraconsistent well-founded model of P is*

$$WFM_p(P) = T \cup not\ \ (\mathcal{H} - \Gamma_s T)$$

This definition is a generalisation of *WFSX* in the following sense:

Theorem 4 (Generalisation of *WFSX*) *Let P be such that $WFM_p(P) = T \cup not\ \ F$. P is non-contradictory iff for no objective literal L, $\{L, \neg L\} \subseteq T$. Moreover, if P is non-contradictory then $WFM_p(P) = WFM(P)$.*

2.2 Fixpoint Approach of Argumentation

The fixpoint approach applied to argumentation, followed by e.g. [Pol87, Pol92, SL92a, Dun93, Dun95, PS97], can be best explained with the idea of reinstatement (cf. Example 1): if an argument A is defeated by an argument B, A can still be acceptable if and only if B is defeated by an argument that is already known to be acceptable. Such an idea is captured by [Dun95]'s notion of acceptability [7], which defines how an argument that cannot defend itself can be protected from attacks by a set of arguments:

Definition 8 (Acceptable Argument) *An argument A is* acceptable *with respect to a set S of arguments iff each argument defeating A is defeated by an argument in S*

[7] Unless indicated otherwise, the other definitions in this section follow from [Dun95].

[Dun95] also defines a characteristic function that returns, for each set S of arguments, the set of all arguments that are acceptable with respect to S. Moreover, the intuitive idea of the characteristic function is that the set of acceptable arguments is constructed step-by-step. First, an empty set is assumed to be the initial set of acceptable arguments. Then, all arguments which are directly acceptable are collected into a set, S^1, by their own strength: these are the ones which are not defeated by any argument. After that, all arguments that are reinstated by arguments in S^1 are added in S^2. More generally, each defeated argument is added in S^{i+1} if it is reinstated by an argument in S^i. This step is repeated until a set S^λ is obtained to which no new argument can be added.

Definition 9 (Characteristic function) *Let $Args$ be a set of arguments, and $S \subseteq Args$. The* characteristic function *F is*

$$F(S) = \{A \in Args \mid A \text{ is acceptable w.r.t. } S\}$$

[Dun95] proves that the characteristic function F is monotonic, and so it has a least fixpoint. In such a case, if an argument is acceptable with respect to S, it is also acceptable with respect to any superset of S. Furthermore, the even cycle (illustrated in Example 2) is avoided by stating that the set of acceptable arguments is the least fixpoint of F. In Example 2, the sets $\{A\}$ and $\{B\}$ are fixpoints of F but none of them is a least fixpoint of F, which is the empty set. In general, we might say that $F(\emptyset) = \emptyset$, if all arguments in the set of arguments are defeated. Based on the notion of the least fixpoint, [Dun95] proposes a skeptical semantics as follows:

Definition 10 (Grounded Extension) *The* Grounded extension *is the least fixpoint of F*

The idea of the least fixpoint is also captured by [PS97] and justified arguments are defined as follows:

Definition 11 (Justified Argument) *An argument is* justified *iff it is a member of the least fixpoint of F*

Proposition 5 *Consider the following sequence of arguments.*

- $F^0 = \emptyset$
- $F^{i+1} = \{A \in Args \mid A \text{ is acceptable with respect to } F^i\}$

Then the following observations holds [Dun95]:

1. *All arguments in* $\bigcup_{i=0}^{\infty}(F^i)$ *are justified*

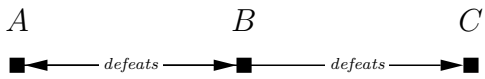
2. *If each argument is defeated by at most a finite number of arguments, then an argument is justified iff it is in* $\bigcup_{i=0}^{\infty}(F^i)$

A peculiarity of Definition 11 is that a distinction between arguments that are non justified is allowed. Then, [PS97] also defines two intermediate statuses for non-justified arguments.

Definition 12 (Overruled and Defensible Arguments) *An argument is* overruled *iff it is not justified, and it is defeated by a justified argument. An argument is* defensible *iff it is neither justified nor overruled*

The decision on how to deal with defeasible arguments is quite controversial. The well-known semantics for non-monotonic reasoning [PP90, vRS91, PA92] conclude that a self-defeating argument (and every argument defeated by it) is defensible. In Example 4, argument B is defeated by A which is a non-justified argument because it is defeated by B (i.e. there is an even cycle between A and B). Since there is no justified argument defeating one of them, both arguments are defeasible. Furthermore, C is defeasible because it is defeated by a non-justified argument. Other solutions are possible, e.g. both [PS97] and [Vre97] distinguish a special 'empty' argument which is not defeated by any other argument and, by definition, defeats any self-defeating argument. In such a case, the argument C of Example 4 is justified because an empty argument reinstates it; A and B are still defensible.

Example 4 (Zombie arguments) *Consider three arguments, A, B and C such that A defeats B, B defeats A, and B defeats C. Neither of the three are justified. B is considered a* zombie argument *because B is neither 'alive' (i.e. justified) nor 'fully dead' (i.e. overruled); it has an intermediate status (i.e. defensible) in which it can still influence the status of other arguments.*



So far, we have presented the fixpoint approach and the grounded (skeptical) extension proposed by [Dun95]. Our proposal is based upon such a grounded extension. However, Dung has also defined three other extensions, viz. stable extension, preferred extension and complete extension. Since such extensions are very well

accepted by the scientific community, we briefly present them here. These extensions are based on the notion of admissible set. Intuitively, such a set represents an admissible, or defendable, point of view. It means that the set of all arguments accepted is a set S of arguments which can defend itself against all attacks on it. Furthermore, it is based on the presumption that S is conflict-free:

Definition 13 (Conflict-free Set) *A set of arguments S is* conflict-free *iff there is no argument in S that defeats an argument in S*

Definition 14 (Admissible Set) *A conflict-free set S of arguments is* admissible *if each argument in S is acceptable w.r.t. S*

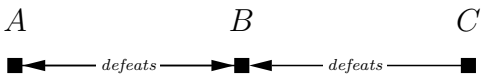
In Example 1, the sets $\emptyset$, $\{C\}$ and $\{A, C\}$ are admissible but all other subsets of $\{A, B, C\}$ are not.

Definition 15 (Stable Extension) *A conflict-free set S of arguments is a stable extension iff every argument that is not in S, is defeated by some argument in S*

In Example 1, the set of arguments $\{A, C\}$ is the only stable extension. Since a stable extension is conflict-free, it reflects in some sense a coherent point of view, i.e. each possible argument is either accepted or rejected. In fact, a stable extension defeats every argument not belonging to it, whether or not that argument is hostile to the extension. Thus, Dung concludes that stable extension does not capture the intuitive semantics of every meaningful argumentation system. However, the preferred extension exists for every argumentation framework:

Definition 16 (Preferred Extension) *A* preferred extension *is a maximal (w.r.t. set inclusion) admissible set of arguments*

Example 5 *Consider three arguments, A, B and C such that A defeats B and vice-versa, and C defeats B. The admissible sets w.r.t. $\{A, B, C\}$ are $\emptyset$, $\{A\}$, $\{B\}$, $\{C\}$ and $\{A, C\}$. The only preferred extension is $\{A, C\}$.*



The stable extension is more skeptical than the grounded extension, the preferred extension is more credulous than the grounded extension, and the complete extension provides a link between credulous and skeptical semantics, i.e. between preferred and ground extensions. [Dun95] proves that each preferred extension is a least complete extension, and the grounded extension is a least (w.r.t. set of inclusion) complete extension.

Definition 17 (Complete Extension) *An admissible set S of arguments is a complete extension iff each argument which is acceptable w.r.t. S, belongs to S*

Preferred and stable extensions are an instance of the multiple-status-assignment approach. The unique-status-assignment approach is also explored with the notion of a grounded extension, already presented above. [DMT02a] has understood non-monotonic reasoning as extending theories in some monotonic language by means of sets of assumptions, provided they are 'appropriate' with respect to some requirements. These are expressed in argumentation-theoretic terms, as follows. According to the semantics of admissible extensions, a set of assumptions is deemed 'appropriate' iff it does not attack itself and it attacks all sets of assumptions which attack it. According to the semantics of preferred extensions, a set of assumptions is deemed 'appropriate' iff it is maximally admissible, with respect to a set of inclusion. According to the semantics of stable semantics, a set of assumptions is deemed 'appropriate' iff it does not attack itself and it attacks every assumption which it does not belong. Given any such semantics of extensions, credulous and skeptical non-monotonic reasoning are defined as follows. A given sentence in the underlying monotonic language is a credulous non-monotonic consequence of a theory iff it holds in some extension of the theory that is deemed 'appropriate' by the chosen semantics. It is a sceptical non-monotonic consequence iff it holds in all extensions of the theory that are deemed 'appropriate' by the chosen semantics.

2.3 Argumentation for Logic Programs

[Dun95] and [PS97] use argumentation to give a declarative semantics for logic programs. Dung says that logic programming with negation as failure can be viewed as a special form of argumentation. The results of his proposal show that logic programming is a good tool for implementing argumentation systems, e.g. [AP96, PS97, BDKT97, dAMAS97, dAMA98a, SPR98, SS02b, PV02, DMT02b]. [PS97] follows Dung's idea but the declarative semantics is refined and a *status for arguments* is defined, viz. justified, overruled or defensible. The basic idea of both proposals is, based on a logic program, to build the set of arguments and so to define the attack relation between those arguments. [Dun95] also shows that argumentation itself can be "viewed" as logic programming by introducing a general method for generating meta-interpreters for argumentation systems. Instead, [PS97] has a proof proposal for such a semantics based on "dialogue trees". The former is a generalized proposal, but the latter goes into detail and so it is easier to develop a prototype of self-argumentation. We will present only the latter.

2.3.1 Dung's Argumentation Framework

This section presents [Dun95]'s grounded semantics. The general idea is that an argument for a certain proposition is a defeasible proof of that proposition in the logic of an underlying language. In this case, the proposition is seen as an objective literal in an extended logic program P (see Def. 2).

Definition 18 (Defeasible Proof) *A* defeasible proof *of an objective literal L is a sequence $r_0; r_1; \ldots; r_n$ of ground rules of an extended logic program P such that*

- $head(r_n) = L$
- *for all i, $0 \leq i \leq n$, if L' is an objective literal in the body of r_i, then there is a $j < i$ such that $L' = head(r_j)$*

The set of default literals supporting a proof for an objective literal L is defined as an argument for L.

Definition 19 (Argument) *An* argument *is a set of ground default literals. Let L be an objective literal and $r_0; \ldots; r_n$ be a defeasible proof for L. A set $A \subseteq \{not\ \ L' \mid not\ \ L' \in Body(r_i)\}$ is an* argument *for L*

Definition 20 (Support) *An* argument *A is a* support *for an objective literal L iff all default literals in a defeasible proof of L are contained in A*

Furthermore, every ground literal L has an argument of the form $(\{not\ L\}, not\ L)$, which captures the idea that L would be concluded *false* if there is no acceptable argument supporting L; otherwise, L is *true*. Finally, the set of all arguments of P is called *Argumentation set of P*.

Definition 21 (Argumentation set) *Let P be an ELP and L an objective literal, then*

$$\begin{aligned} AR(P) = \ & \{(A, L) \mid A \text{ is an argument for } L\} \ \cup \\ & \{(\{not\ \ L\}, not\ \ L\}) \mid L \text{ is a ground atom}\} \end{aligned}$$

is the argumentation set *of P*

An argument A is *sound* if there is no objective literal L such that A supports both L and $\neg L$; otherwise, it is *self-defeating*. Based on such notions, viz. sound and self-defeating, two kinds of attack are defined: *RAA*-attack (or *Reductio Ad Absurdum*-attack) and *g*-attack (or *ground*-attack). An argument attacks another argument via *RAA*-attack if both arguments together support an objective literal and its explicit negation. Stronger than this is a *g*-attack because it refutes a given argument directly: an argument A_1 is *g*-attacked by an argument A_2 if A_2 is an argument for L and there is a default literal *not* L in A_1.

Definition 22 (*RAA*- and *g*-attack) *Let A_1 and A_2 be sound arguments. The argument A_2 is a RAA-attack against A_1 (and vice-versa) if $A_1 \cup A_2$ is self-defeating; and A_2 is a g-attack against A_1 if there is an assumption 'not L' in A_1 such that L is supported by A_2.*

The argumentation framework of program P is presented in the following definition. Then, Example 6 illustrates [Dun95]'s argumentation proposal.

Definition 23 (Argumentation framework) *Let P be an ELP, AR be the set of arguments of P, and attacks $\subseteq AR \times AR$, then $AF(P) =< AR, attacks >$ is called an* argumentation framework.

Example 6 *Let $P = \{a \leftarrow not\ b;\ b \leftarrow not\ a;\ c \leftarrow not\ b;\ \neg a; d \leftarrow not\ e; e\}$. The sequence 'a $\leftarrow$ not b' is a defeasible proof for a, and $\{not\ b\}$ is an argument for a. Since $\neg a$ is a fact in P, a defeasible proof for $\neg a$ is the fact itself and so the empty set is an argument for $\neg a$. The argumentation set of P is*

$$\begin{aligned} AR(P) = \{ & (\emptyset, \neg a), (\{not\ a\}, b), (\{not\ b\}, a), (\{not\ b\}, c), (\{not\ e\}, d), (\emptyset, e) \ \} \cup \\ \{ & (\{not\ \neg a\}, not\ \neg a), (\{not\ b\}, not\ b), (\{not\ a\}, not\ a), \\ & (\{not\ c\}, not\ c), (\{not\ d\}, not\ d), (\{not\ d\}, not\ d) \ \} \end{aligned}$$

Figure 2.1 illustrates the attacking relation between arguments in $AR(P)$. Conforming to definitions of Fixpoint operator F of the argumentation framework $AF(P)$ and Grounded Extension (Def. 9 and Def. 10 in Section 2.2, respectively), we obtain the following:

- $S^0 = \emptyset$
- $S^1 = F(S^0) = \{(\emptyset, e)\}$
- $S^2 = F(S^2) = \{(\emptyset, e), (\{not\ d\}, not\ d)\}$
- $S^3 = F(S^2)$

So, S^3 is the least fixpoint of F, and the set of acceptable arguments of $AR(P)$. The arguments in S^3 are then justified (cf. Def. 11 in Section 2.2). Moreover, e is true and d is false.

2.3.2 Prakken and Sartor's Argumentation Framework

[PS97]'s proposal is somewhat different from the previous one. An *argument* is seen as a sequence of rules that can be chained together, and it is grounded on the facts.

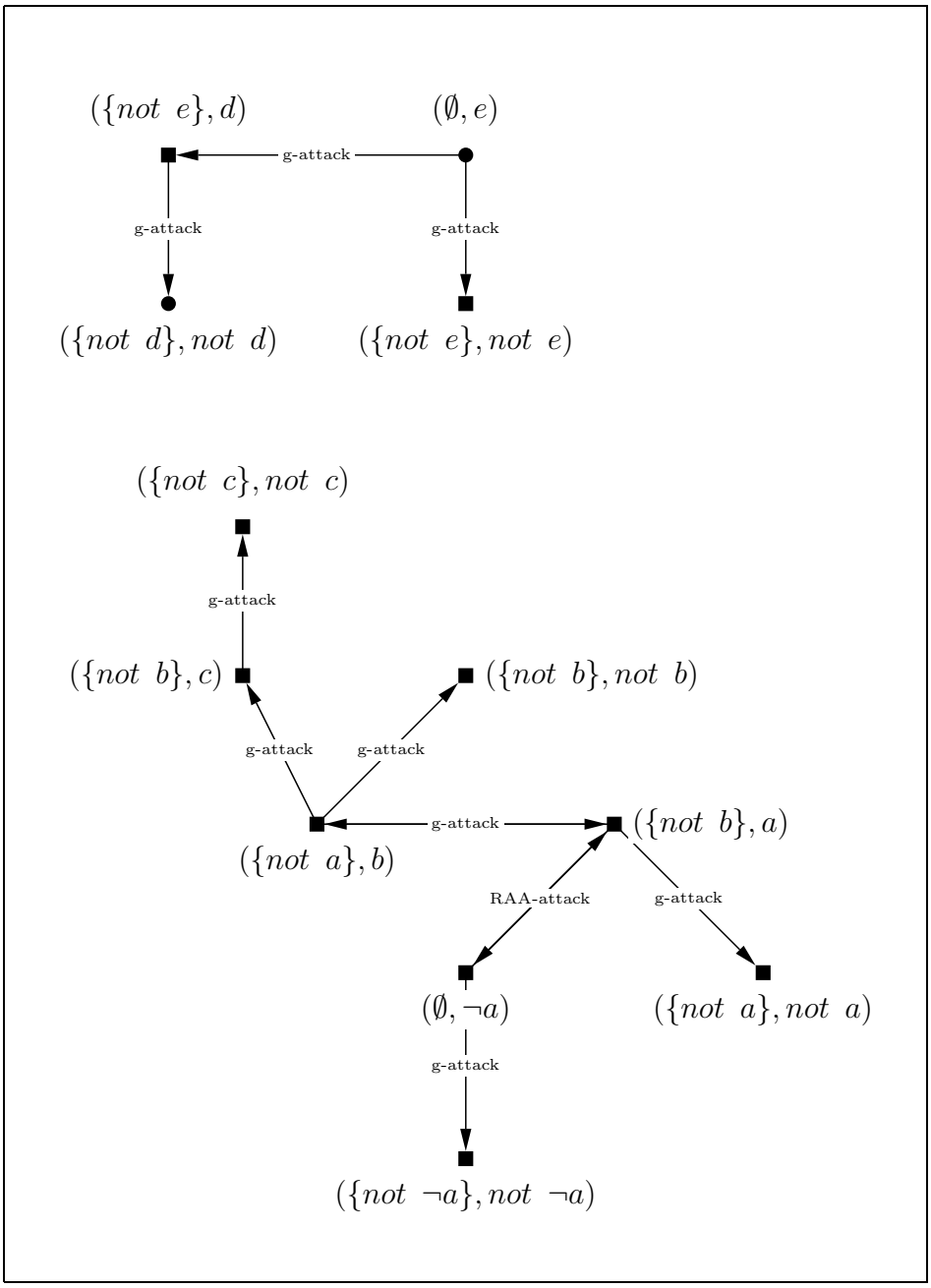


Figure 2.1: Attacking relation of Example 6

Definition 24 (Argument and Sub-argument) *Let P be an ELP. An* argument *for a conclusion L is a finite sequence $A = [r_n; \ldots; r_m]$ of rules $r_i \in P$ such that*

- *for every i $(n \leq i \leq m)$, and for every objective literal L_j in the body of r_i there is a $k < i$ such that L_j is the head of r_k*
- *L is the head of some rule of A*
- *No two distinct rules in the sequence have the same head*

An argument A' (for some conclusion L') is a sub-argument *of the argument A (possibly for some other conclusion L) iff A' is a subset of A.*

An argument attacks another argument via *rebut* or *undercut*. The difference depends on whether the attacking argument contradicts a conclusion or an assumption of another argument.

Definition 25 (Undercut, Rebut, Attack) *Let A_1 and A_2 be arguments, then*

- A_1 undercuts A_2 *iff (i) A_1 is an argument for L and (ii) A_2 is an argument with assumption not L, i.e. there is an $r : L_0 \leftarrow L_1, \ldots, L_l$, not $L_{l+1}, \ldots,$ not $L_m \in A_2$ and a j $(l+1 \leq j \leq m)$ such that $L = L_j$;*
- A_1 rebuts A_2 *iff (i) A_1 is an argument for L and (ii) A_2 is an argument for $\neg L$;*
- A_1 attacks A_2 *iff A_1 undercuts or rebuts A_2.*

The notions of *coherent argument* and *conflict-free set of arguments* deal with the self-defeating problem:

Definition 26 (Coherent, Conflict-free) *An argument is* coherent *if it does not contain sub-arguments attacking each other. A set of arguments Args is called* conflict-free *if no two arguments in Args attack each other.*

Defeat of an argument can be direct, or indirect, by defeating one of its sub-arguments. In particular, any incoherent argument is defeated by an empty argument.

Definition 27 (Defeat, Strictly Defeat) *Let A_1 and A_2 be two arguments. A_1* defeats *A_2 iff (i) A_1 is empty and A_2 incoherent, or (ii) A_1 undercuts A_2 or A_1 rebuts A_2 and A_2 does not undercut A_1. A_1* strictly defeats *A_2 iff A_1 defeats A_2 but not vice versa.*

Note that restriction (ii) of the above definition allows that an undercutting attack is stronger than a rebutting attack. For instance, consider the following two rules:

$$innocent \leftarrow not\ \neg innocent.$$
$$\neg innocent.$$

Although argument $A = [innocent \leftarrow not\ \neg innocent]$ rebuts $B = [\neg innocent]$, A does not defeat B since B undercuts A. So, B strictly defeats A. [PS97] imposes such a restriction motivated by the legal principle that "the law should be interpreted as coherently as possible".

The definition of acceptable argument is quite different from Definition 8 in Section 2.2:

Definition 28 (Acceptable) *An argument A is* acceptable *w.r.t. a set of arguments Args iff each argument defeating A is strictly defeated by an argument in Args.*

This proposal follows the fixpoint operator [Dun93], which captures the set of acceptable arguments:

Definition 29 (Characteristic Function) *Let P be an ELP and S be a subset of arguments of P. The* characteristic function *of P and S is*

$$F_P(S) = \{A \in S \mid A \text{ is acceptable w.r.t. } S\}$$

The conclusion of the status of arguments is based on the ways in which they interact. The characteristic function takes as input the set $Args$ of all possible arguments and their mutual relations of defeat, and produces as output the set of acceptable arguments w.r.t. $Args$. Then all arguments in $Args$ split into three classes:

Definition 30 (Justified, Overruled, Defensible) *Let P be an ELP and F_P be the* characteristic function *of P then A is* justified *iff A is in the least fixpoint of F_P (called $JustArgs$); A is* overruled *iff A is not justified and it is attacked by a justified argument; and A is* defensible *iff A is neither justified nor overruled.*

The proposal requires that $JustArgs$ is conflict-free; otherwise, every argument (including the empty argument) attacks itself.

Proposition 6 *The set of justified arguments is conflict-free. Otherwise, every argument is defensible for any F_P.*

Definition 31 (Conclusion) *For any literal L, L is a* justified conclusion *iff it is a conclusion of a justified argument; a* defensible conclusion *iff it is not justified and it is a conclusion of some defensible argument; and an* overruled conclusion *iff it is not justified nor defensible, and is a conclusion of an overruled argument.*

Example 7 illustrates [PS97]'s argumentation framework.

Example 7 *Let $P = \{a \leftarrow not\ \ b;\ b \leftarrow not\ \ a;\ c \leftarrow not\ \ b;\ \neg a;\ d \leftarrow not\ \ e;\ e\}$. The argumentation set $Args$ of P is*

$$Args = \{\ \ [a \leftarrow not\ \ b],\ [b \leftarrow not\ \ a],\ [c \leftarrow not\ \ b],\ [\neg a],\ [d \leftarrow not\ \ e],\ [e]\ \ \}$$

or $Args = \{A_a, A_b, A_c, A_{\neg a}, A_d, A_e\}$. The Figure 2.2 illustrates both defeat and

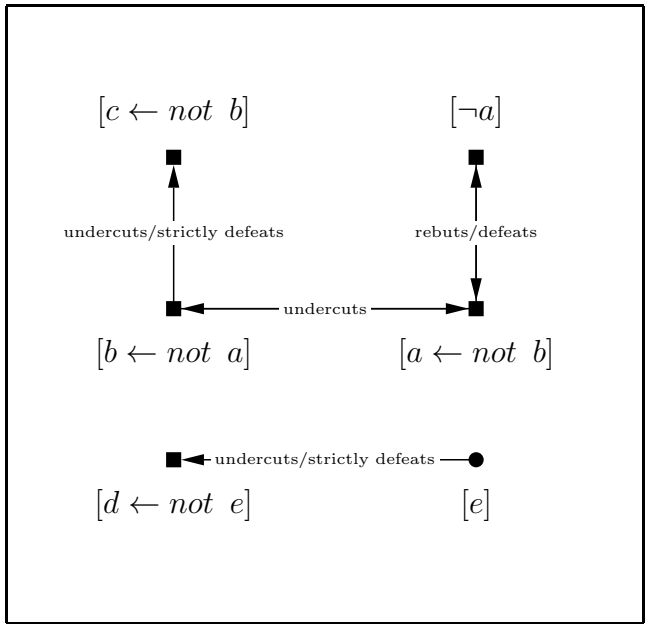


Figure 2.2: Defeating and Strictly defeating relation in $Args$

strictly defeat relation between arguments in $Args$. We then obtain the following results from $F_P(\emptyset)$:

- $S^0 = \emptyset$
- $S^1 = F(S^0) = \{A_e\}$
- $S^2 = F(S^1)$

S^2 is the least fixpoint of $F_P(\emptyset)$ and so it is the set of acceptable arguments w.r.t. $Args$. Thus, A_e is a justified argument, A_d is an overruled argument and the defensible arguments are A_b, A_c, A_a and $A_{\neg a}$. Furthermore, e is a justified conclusion, d is an overruled conclusion and the defensible conclusions are b, c, a and $\neg a$.

[PS97] also formalises an argumentation process using extended logic programs augmented with priorities, by extending [Dun93]'s grounded semantics to incorporate such a priorities. In alternative, [PS97] defines a more credulous semantics in which defensible arguments can be defended together. In both proposals, it is assumed that there is a fixed and undisputed ordering of the rules. Since we will not deal explicitly with preferences rules, we do not present that proposal.

A proof for an argument

A proof for an argument is a dialogue tree where the root of the tree is an argument for L, and each branch of the tree is a dialogue between a proponent P and an opponent O. A *move* in a dialogue consists of an argument attacking the last move of the other player. The required strength of a move depends on who states it. Since the proponent wants a conclusion to be justified, a proponent's argument has to be strictly defeating. The opponent simply wants to prevent the conclusion from being justified. Thus there is no need for its move to be strictly defeating; it is enough for it to be defeating.

Definition 32 (Dialogue) *A* dialogue *is a finite nonempty sequence of moves $move_i = (Player_i, A_i)(i > 0)$, such that*

1. *$Player_i = P$ iff i is* odd*; and $Player_i = O$ iff i is* even
2. *If $Player_i = Player_j = P$ and $i \neq j$, then $A_i \neq A_j$*
3. *If $Player_i = P(i > 1)$, then A_i is a minimal (w.r.t. set inclusion) argument strictly defeating A_{i-1}*
4. *If $Player_i = O$, then A_i defeats A_{i-1}*

The first condition says that P begins and the players take turns. The second condition prevents the proponent from repeating its attacks. The remaining two conditions form the core of the definition: they state the burdens of proof for P and O. The minimally condition on P's move makes it impossible to make arguments trivially different by combining them with some other, irrelevant argument.

A dialogue tree considers all possible ways in which an opponent can attack an argument:

Definition 33 (Dialogue Tree) *A* dialogue tree *is a finite tree of moves* $move_i = (Player_i, A_i)$, $Player_i \in \{P, O\}$ *such that*

1. *Each branch is a dialogue*

2. *If* $Player_i = P$ *then the children of* $move_i$ *are all defeaters of* A_i

A player wins a dialogue tree iff it wins all branches (i.e. dialogues) of the tree and a player wins a dialogue if the other player cannot move (i.e. counter-argue).

This definition also marks dialogues tree candidates for being proofs: it says that the tree should consider all possibly ways in which O can attack an argument P.

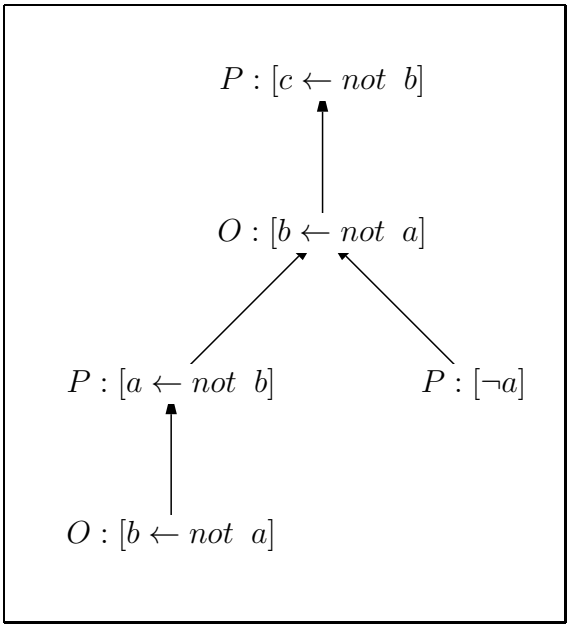
Definition 34 *An argument is* provably justified argument *iff there is a dialogue tree with A as its root, and won by the proponent. And a strong literal L is* provably justified conclusion *iff it is a conclusion of a provably justified argument.*

Proposition 7 *All provably justified arguments are justified.*

Proposition 8 *For finitary ordered theories each justified argument is provably justified.*

Proposition 9 *If all argument is provably justified, then all its subarguments are provably justified.*

Example 8 *Let P be the program in Example 7. The figure below illustrates a dialogue tree DT for* $[c \leftarrow not\ \ b]$. *The proponent player does not win DT because there is a last movement* $(O, [b \leftarrow not\ \ a])$ *in the first dialogue (from left to right side) that cannot be counter-attacked. Thus,* $[c \leftarrow not\ \ b]$ *is not justified. We should then evaluate* $(O, [b \leftarrow not\ \ a])$ *to determine if it is justified or not. Note that the argument* $[b \leftarrow not\ \ a]$ *occurs twice in the first dialogue as the opponent's moves. We can say the second move is to reinstate the argument itself. So, the argument for c is not justified.*


$P : [c \leftarrow not\ \ b]$
$O : [b \leftarrow not\ \ a]$
$P : [a \leftarrow not\ \ b]$
$P : [\neg a]$
$O : [b \leftarrow not\ \ a]$

Chapter 3

A Proposal for Self-Argumentation

This chapter presents an argumentation semantics which involves a "single" extended logic program, named self-argumentation *semantics. We focus on the properties of a declarative semantics in what regards paraconsistency which are interesting by themselves, and independent from its distributed nature. With this purpose, we restrict our attention to the special case of the distributed semantics, where only a "single" extended logic program with denials (ELPd) is in the set of programs. The* self-argumentation *semantics is inspired by two well known argumentation semantics, viz. [Dun95] and [PS97], overviewed in the previous chapter. We redefine [PS97]'s definition of argument and other [PS97]'s definitions are simplified. The goal being to obtain a semantics for extended logic program with denials which represents the knowledge base of an agent. Furthermore, we propose a parameterized characteristic function so that our argumentation semantics obtains different levels of acceptability of an argument. With such a differentiation, we go through the properties of both conflict-free and contradictory sets of acceptable arguments. Therefore, we obtain both paraconsistent and consistent ways of reasoning. According to [PS97]'s definition of the status of an argument, the argument may be justified, overruled or defeasible. On top of that, we propose that a justified argument can be contradictory, based on contradiction, or non-contradictory. We then present a definition of the truth value of a conclusion G such that G is true (and contradictory, based-on-contradiction, or non-contradictory), false or undefined. Finally, we present a proof procedure for such a declarative semantics.*

In logic programming, several ways to formalize argumentation-based semantics have been studied for a single logic program (e.g. [RS09, arg10], and scientific events such as "Conference on Computational Models of Argument (COMMA)", "Conference on Principles of Knowledge Representation and Reasoning" (KR), "Argument, Dialog and Decision" on International workshop on Non-Monotonic Reasoning (NMR) and "Workshop Argumentation and Non-Monotonic Reasoning" (ArgNMR)). Intuitively, argumentation-based semantics treat the evaluation of a logic program as an argumentation process, i.e. a goal G is true if at least one argument for G cannot be successfully attacked. The ability to view logic programming as a non-monotonic knowledge representation language, in equal standing with other non-monotonic logics, brought to light the importance of defining clear declarative semantics for logic programs, for which proof procedures (and attending implementations) are then defined (e.g. [Dun93, Dun95, PS97, BDKT97, Vre97, Lou98, SS02b, DMT02a, Pol01, DMT02b, GS04, Pra09]), as overviewed in the previous chapter.

The main goal of the thesis is to propose an argumentation-based semantics for sets of logic programs that are able to cooperate and argue with each other, named distributed semantics. In it each program relies on a set of other programs with which it has to agree in order to accept an argument, and a set of programs with which it can cooperate to build arguments. Besides this distributed nature, the distributed semantics also allows for paraconsistent forms of argumentation. In fact, it is also a goal of this proposal to be able to deal with mutually inconsistent, and even inconsistent, knowledge bases. Moreover, when in presence of contradiction, we want to obtain ways of agent reasoning, ranging from consistent (in which inconsistencies lead to no result) to paraconsistent. For achieving this, we consider strong and weak arguments. The paraconsistency in the argumentation also yields a refinement of the possible status of arguments: besides the justified, overruled, and defensible arguments as in [PV02], justified arguments may now be contradictory, based on contradiction, or non-contradictory. Moreover, in some applications it might be interesting to change easily from a paraconsistent to a consistent way of reasoning (or vice-versa).

In this chapter we focus on the properties of that declarative semantics in what regards paraconsistency which are interesting by themselves, and independent from its distributed nature. With this purpose, we restrict our attention to the special case of the distributed semantics where only a single logic program is in the set of programs, i.e. we propose a semantics for an extended logic program with denials (ELPd) (see Def. 3) which represents the knowledge base of an agent. The semantics is argumentation-based, in the line of the work developed by [Dun95, PS97] for defining semantics of single extended logic programs. As described in Chapter 2, in these argumentation-based semantics, the rules of a logic program are viewed

as encoding arguments of an agent. Therefore, the basic notion of argumentation systems is not that of a defeasible conclusion, but that of a defeasible argument for this conclusion. By defeasible conclusion (or argument) we mean that it "is reasonable" and it is supported by some sort of argumentation process. Although the construction of arguments is monotonic, i.e. arguments remain if more rules are added to the program, in practice, the defeasibility is explained in terms of the interactions between conflicting arguments. Non-monotonicity arises from the fact that new conclusions may give rise to stronger counter-arguments, which might defeat previously built arguments. Moreover, the truth value of a conclusion is determined by whether its arguments, depending on the specific semantics, can or cannot defend themselves from the attacks of other arguments. In the remainder of this chapter we first motivate and illustrate our proposal with an example. Then we define a declarative semantics for self-argumentation and, after that, we present a proof procedure for such a declarative semantics. Finally, some conclusions are presented.

3.1 "Privacy and Personal Life", an example

We are going to motivate and illustrate our proposal of self-argumentation for an ELPd with the help of the following example. It describes, in an informal way, how arguments are built from an ELPd that models a knowledge base of an agent.

Example 9 (Privacy of Personal Life – PPL) *Usually, any person deserves privacy with respect to her personal life. However, when such a person behaves in a way that is not acceptable (e.g. selling drugs), she will suffer the consequences. The first consequence is the focus of media attention on her personal life and consequent loss of privacy. The personal life of such a person might be exposed by the "results" of media attention (e.g. photos, reports, and so on) when there is no law that protects her against it. The above description can be expressed by the following extended logic programming rules.*

$$focusOfMediaAttention(X) \leftarrow person(X),\ \neg acceptableBehavior(X).$$
$$\neg acceptableBehavior(X) \leftarrow event(X,Y),\ againstSociety(Y).$$
$$\neg hasPrivacy(X) \leftarrow focusOfMediaAttention(X).$$
$$personalLifeExposed(X) \leftarrow \neg hasPrivacy(X), not\ protectedByLaw(X).$$
$$hasPrivacy(X) \leftarrow person(X),\ not\ \neg hasPrivacy(X).$$

In contrast, it is considered an absurdity that someone may lose her privacy when she is involved in some event for which there is no evidence that it should be made public (e.g. someone starting a long-term treatment for drugs dependency).

The absurdity in the rule below is represented as a denial, and the symbol $\bot$ denotes it (cf. Def. 3).

$$\bot \leftarrow \neg hasPrivacy(X),\ event(X,Y),\ not\ \ publicEvent(Y).$$

Moreover, modern society normally tries to protect children, and so their privacy is guaranteed until evidence appears of some unusual behavior (e.g. by having unacceptable behavior).

$$hasPrivacy(X) \leftarrow child(X),\ not\ \ unusualChild(X).$$
$$unusualChild(X) \leftarrow child(X), \neg acceptableBehavior(X).$$
$$person(X) \leftarrow child(X).$$

However, famous persons are inherently the focus of media attention:

$$focusOfMediaAttention(X) \leftarrow famousPerson(X).$$
$$person(X) \leftarrow famousPerson(X).$$

Assume an agent Ag with the knowledge above, plus some facts about three persons: Apuã, Poti, and Ivoti [1] *such that Ag knows that Poti is a famous child, Apuã was seen selling drugs — a criminal behavior against society —, and Ivoti is a famous soccer player in treatment for drugs dependency:*

$$child(poti).$$
$$famousPerson(poti).$$
$$person(apua).$$
$$event(apua, sellsDrugs).$$
$$againstSociety(sellsDrugs).$$
$$famousPerson(ivoti).$$
$$event(ivoti, treatmentForDrugsDependency).$$

Figure 3.1 illustrates, with obvious abbreviations, the set of rules PPL which will be used for illustration in the remainder of this chapter. Figure 3.2 simply clarifies the notation of the objective literals over PPL.

Following [PS97]'s definition, an argument for an objective literal A is a sequence of rules that "proves" A if all default literals (of the form *not* B) in the body of those rules are assumed true. For instance, the following sequences of rules are arguments of Ag for conclusions related to Poti over PPL. Each argument is presented in the form "$L - A_L : [r_{L'}; \ldots; r_L]$", which means "a conclusion L is supported by an argument A_L composed by the sequence of rules $[r_{L'}; \ldots; r_L]$". Furthermore, "$L - A_L : A_{L''} + [r_{L'}; \ldots; r_L]$" means "an argument A_L is built based on some previous argument $A_{L''}$".

[1] The following names are from Native South Americans, more specifically from the Tupi-Guarani family. Apuã [*apu'a*] means "typhoon", and Poti and Ivoti [*poty* and *yvoty*] both mean "flower". For details see *http://en.wikipedia.org/wiki/Tupi_people*.

$$PPL = \left\{ \begin{array}{l} fOMA(X) \leftarrow pe(X), \neg acB(X); \\ \neg acB(X) \leftarrow ev(X,Y), aS(Y); \\ \neg hP(X) \leftarrow fOMA(X); \\ pLE(X) \leftarrow \neg hP(X), not\ \ pBL(X); \\ hP(X) \leftarrow pe(X), not\ \ \neg hP(X); \\ \bot \leftarrow \neg hP(X), ev(X,Y), not\ \ pE(Y); \\ hP(X) \leftarrow ch(X), not\ \ uC(X); \\ uC(X) \leftarrow ch(X), \neg acB(X); \\ pe(X) \leftarrow ch(X); \\ fOMA(X) \leftarrow fP(X); \\ pe(X) \leftarrow fP(X); \\ ch(p); \quad fP(p); \\ pe(a); \quad ev(a, sD); \quad aS(sD); \\ fP(i); \quad ev(i, tFDD) \end{array} \right\}$$

Figure 3.1: The knowledge of agent Ag about "Privacy of Personal Life"

Poti is a child – $A_{ch(p)} : [ch(p)]$

Poti is a famous person – $A_{fP(p)} : [fP(p)]$

Poti is a person –
$A_{pe(p)} : A_{ch(p)} + [pe(p) \leftarrow ch(p)]$
$A'_{pe(p)} : A_{fP(p)} + [pe(p) \leftarrow fP(p)]$

Poti is the focus of media attention –
$A_{fOMA(p)} : A_{fP(p)} + [fOMA(p) \leftarrow fP(p)]$

Poti has no privacy – $A_{\neg hP(p)} : A_{fOMA(p)} + [\neg hP(p) \leftarrow fOMA(p)]$

Poti has her personal life exposed –
$A_{pLE(p)} : A_{\neg hP(p)} + [pLE(p) \leftarrow \neg hP(p), not\ \ pBL(p)]$

Poti has privacy –
$A_{hP(p)} : A_{pe(p)} + [hP(p) \leftarrow pe(p), not\ \ \neg hP(p)]$
$A'_{hP(p)} : A_{ch(p)} + [hP(p) \leftarrow ch(p), not\ \ uC(p)]$

[PS97]'s argumentation semantics determines the acceptability of arguments based on certain definitions, viz. *attack*, *defeat*, *strictly defeat* and *acceptable argument* (for details, see Section 2.3.2). Differently, we introduce a new kind of argument and so simplify some of these definitions. We call the above arguments strong arguments, and we propose a *weak version of a strong argument*. To distinguish them, a strong argument for L will be denoted by A^s_L and its weak version by

$ch(p)$	"Poti is a child"
$pe(p)$	"Poti is a person"
$hP(p)$	"Poti has privacy"
$fP(p)$	"Poti is a famous person"
$\neg hP(p)$	"Poti has no privacy"
$fOMA(p)$	"Poti is the focus of media attention"
$pLE(p)$	"Poti has her personal life exposed"
$pe(a)$	"Apuã is a person"
$ev(a, SD)$	"Apuã is involved in selling drugs"
$\neg acB(a)$	"Apuã has unacceptable behaviour"
$fOMA(a)$	"Apuã is the focus of media attention"
$\neg hP(a)$	"Apuã has no privacy"
$pLE(a)$	"Apuã has his personal life exposed"
$aS(SD)$	"Selling drugs is against society"
$hP(a)$	"Apuã has privacy"
$fP(i)$	"Ivoti is a famous person"
$fOMA(i)$	"Ivoti is the focus of media attention"
$\neg hP(i)$	"Ivoti has no privacy"
$pLE(i)$	"Ivoti has his personal life exposed"
$ev(i, tFDD)$	"Ivoti is in treatment for drugs dependency"
$pe(i)$	"Ivoti is a person"
$hP(i)$	"Ivoti has privacy"

Figure 3.2: The conclusions over the set of rules PPL

A^w_L. For every rule for L (denoted by r_L) from A^s_L, A^w_L is built by adding $not\ \neg L$ and $not\ \bot$ in r_L, thus making the rules weaker (more susceptible to being contradicted/attacked). Intuitively, if there is a potential inconsistency, be it by proving the explict complement of a rules head or by proving $\bot$, then the weak argument is attacked, whereas the strong is not. For instance, the arguments below are the weak version of $A^s_{\neg hP(p)}$, $A^s_{hP(p)}$ and $A'^s_{hP(p)}$, respectively.

Poti has no privacy –

$$A^w_{\neg hP(p)} : A^w_{fOMA(p)} + [\neg hP(p) \leftarrow fOMA(p), not\ \ hP(p), not\ \ \bot]$$

Poti has privacy –

$$A^w_{hP(p)} : A^w_{pe(p)} + [hP(p) \leftarrow pe(p), not\ \ \neg hP(p)^2, not\ \ \bot]$$
$$A'^w_{hP(p)} : A^w_{ch(p)} + [hP(p) \leftarrow ch(p), not\ \ uC(p), not\ \ \neg hP(p), not\ \ \bot]$$

For simplicity, not every argument involved is show, viz. $A^w_{fOMA(p)}$, $A^w_{pe(p)}$ and $A^w_{ch(p)}$. Assume they are built in the same way as the above arguments.

Finally, we showed that an argument of an agent Ag for L is

- a sequence of ground rules started by a rule r with L in its head (i.e. $r = L \leftarrow Body$);
- the rules are chained together and based on facts; and
- all negative literals $not\ L'$ in the rules of the resulting sequence are hypotheses that there is no evidence for L' in Ag

Therefore, every $not\ L'$ in the sequence can be attacked by some argument for L', i.e. the hypothesis $not\ L'$ is attacked by the evidence of L'. For instance, the weak arguments for "Poti has privacy" can be attacked by both strong and weak arguments for "Poti has no privacy", i.e. both $A^s_{\neg hP(p)}$ and $A^w_{\neg hP(p)}$ can attack the hypothesis $not\ \neg hP(p)$ in $A^w_{\neg hP(p)}$. A similar reasoning, by both $A^s_{hP(p)}$ and $A^w_{hP(p)}$, they can attack $not\ \ hP(p)$ in $A^w_{\neg hP(p)}$.

3.2 Declarative Semantics

We illustrated and motivated our self-argumentation proposal on the previous section. Now, we present the formal definitions. First, assume that the knowledge base of an agent Ag is an *Extended Logic Program with denials* P over a *language* (cf. Def. 3 and Def. 1, respectively), and an atom L of an agent[3] Ag is in the *Extended Herbrand Base of* the program P (cf. Def. 3).

[2]We suppress the other $not\ \neg hP(p)$ because it is already in the original rule for $hP(X)$.

[3]To simplify the notation, we will refer, in the sequel, to the logic program that represents an agent's knowledge base simply as an "agent".

As already mentioned in the previous section, both strong and weak arguments are sequences of rules. Nevertheless, it is implicit that these sequences of rules should be complete and well-defined. By *complete* we mean that all required rules are in the sequence. By *well-defined sequence* we mean a (minimal) sequence of rules for some L. Since a well-defined sequence is based on a set of rules of an agent Ag, we first define what we mean by *sets of rules of* Ag. A *strong set of rules* is defined for building strong arguments, and it is the set of rules of Ag. The definition of a *weak set of rules* is slightly different. For every rule of L (denoted by r_L) within the set of Ag's rules, a 'weak version' of r_L is built by adding $not\ L$ and $not\ \bot$ in the body of r_L. The resulting set of rules is used for building weak arguments.

Definition 35 (Strong and Weak Sets of Rules) *Let $\mathcal{L}$ be a language, and P be an ELPd over $\mathcal{L}$. The* strong set of rules of *P is*

$$R_P^s = \ P$$

and the weak set of rules of *P is*

$$R_P^w = \ \{\ L \leftarrow Body, not\ \neg L, not\ \bot \mid L \leftarrow Body \in P\ \}$$

We say R_P is a set of rules*, if it is either a strong or a weak set of rules of P.*

Remark 10 *In the remainder, whenever the language is clear from the context, we omit it, and simply say "Let P be an ELPd" instead of "Let $\mathcal{L}$ be a language, and P be an ELPd over $\mathcal{L}$".*

Instead of using an example where rules and predicates have "real meaning", as in Example 9 for motivating the proposal, here we use an example tailored to illustrate the technical details of some of definitions.

Example 10 *Let P be an ELPd as follows*

$$\begin{array}{ll}\{ & a;\ \neg a;\ b \leftarrow a;\ c \leftarrow not\ \ a;\ d \leftarrow not\ \ a, not\ \ e;\ e \leftarrow not\ \ f; \\ & e \leftarrow not\ \ g;\ f;\ g;\ g \leftarrow not\ \ c;\ h \leftarrow not\ \ g;\ i \leftarrow not\ \ j;\ j \leftarrow not\ \ i\ \}\end{array}$$

Then $R_P^s = P$ and R_P^w is

$$\begin{array}{ll}\{ & a \leftarrow not\ \ \neg a, not\ \ \bot;\ \neg a \leftarrow not\ \ a, not\ \ \bot; \\ & b \leftarrow a, not\ \ \neg b, not\ \ \bot;\ c \leftarrow not\ \ a, not\ \ \neg c, not\ \ \bot; \\ & \ldots \\ & i \leftarrow not\ \ j, not\ \ \neg i, not\ \ \bot;\ j \leftarrow not\ \ i, not\ \ \neg j, not\ \ \bot\ \}\end{array}$$

The R_P^w presented above has a subset of the rules of P, since the aim is simply to exemplify the method of rule-building.

A well-defined sequence for an objective literal L is then built as follows: the last rule is a rule for L (i.e. $L \leftarrow Body$) and the previous are rules for the objective literals L_i in $Body$. This procedure is recursive, i.e. for each literal L_i there must exist a rule r for L_i (and so a sequence of rules for each objective literal in the body of r). Furthermore, the sequence is built by chaining rules together, only using those that are strictly necessary and ignoring default literals. Moreover, the sequence must not be circular. Finally, the first rule in the sequence for a complete well-defined sequence should be either a fact or a rule whose body only has default literals.

Definition 36 (Well-defined and Complete Sequence) *Let P be an ELPd, and $L \in \mathcal{H}(P)$. A* well-defined sequence *for L over a set of (ground) rules S is a finite sequence $[r_1; \ldots; r_m]$ of rules r_i from S of the form $L_i \leftarrow Body_i$ such that*

1. *L is the head of the rule r_m, and*
2. *an objective literal L' is the head of a rule r_i ($1 \leq i < m$) only if L' is not in the body of any r_k ($1 \leq k \leq i$) and L' is in the body of some rule r_j ($i < j \leq m$).*

We say that a well-defined sequence for L is complete *if for each objective literal L' in the body of the rules r_i ($1 \leq i \leq m$) there is a rule r_k ($k < i$) such that L' is the head of r_k.*

Example 11 *Assume R_P^s and R_P^w of Example 10, a (non-complete) well-defined sequence for the objective literal b over R_P^s (resp. R_P^w) is $[b \leftarrow a]$ (resp. $[b \leftarrow a, not\ \neg b, not\ \bot]$). A complete well-defined sequence for b over R_P^s (resp. R_P^w) is $[a; b \leftarrow a]$ (resp. $[a \leftarrow not\ \neg a, not\ \bot; b \leftarrow a, not\ \neg b, not\ \bot]$). $[d \leftarrow not\ a, not\ e]$ is a complete well-defined sequence for the objective literal d over R_P^s, and it is only composed by the rule for d (r_d) since there is no objective literal in the body of r_d. A similar situation arises in a complete well-defined sequence for the objective literal a over R_P^s, which is $[a]$. The objective literal e has two complete well-defined sequences over R_P^s, viz. $[e \leftarrow not\ f]$ and $[e \leftarrow not\ g]$; a similar situation for the objective literal g is $[g]$ and $[g \leftarrow not\ c]$.*

Definition 37 (Strong and Weak Arguments) *Let P be an ELPd, $L \in \mathcal{H}(P)$, and R_P^s (resp. R_P^w) be the strong (resp. weak) set of rules of P. A* strong *(resp.* weak*)* argument *of P for L, A_L^s (resp. A_L^w), is a complete well-defined sequence for L over R_P^s (resp. R_P^w).*

Let A_L^w and A_L^s be two arguments of P. A_L^w is the weak argument corresponding *to A_L^s, and vice-versa, if both use exactly the same rules of the original program P (the former by having instances of rules R_P^w and the latter from R_P^s).*

We say that A_L is an argument *of P for L if it is either a strong argument or a weak one of P for L. We also say that A_L^k is a* k-argument of P *for L (where k is either s, for strong arguments, or w, for weak ones).*

Remark 11 *Since this chapter proposes self-argumentation, which always involves a single agent alone, we will say "argument for L" instead of "argument of Ag for L". Furthermore, we also say "weak argument for L" instead of "weak argument corresponding to a strong argument for L".*

Example 12 *Following Example 11, the strong (resp. weak) argument for the objective literal b is the complete well-defined sequence for b over R_P^s (resp. R_P^w). Since the objective literal e has two complete well-defined sequences over R_P^s (resp. R_P^w), the literal e has two strong (resp. weak) arguments.*

The set of arguments of an agent Ag is obtained by building every strong argument, and the corresponding weak one, for every L in Ag's knowledge base (cf. Def. 3). Note that if some objective literal L does not appear in the head of some rule, the well-defined sequence for L is empty and so there is no argument for L.

Definition 38 (Set of Arguments) *Let P be an ELPd. The set of k-arguments of P is*

$$\mathcal{A}rgs^k(P) = \bigcup_{L_i \in \mathcal{H}(P)} A_P^k(L_i)$$

where $A_P^s(L_i)$ (resp. $A_P^w(L_i)$) denotes the set of all strong (resp. weak) arguments of P for L_i. We denote by $\mathcal{A}rgs(P)$ the set of all s-arguments and w-arguments of P, i.e.

$$\mathcal{A}rgs(P) = \mathcal{A}rgs^s(P) \cup \mathcal{A}rgs^w(P).$$

Example 13 *Assume the R_P^s and R_P^w of Example 10. The set of s-arguments of P is*

$$\begin{aligned}\mathcal{A}rgs^s(P) = \{\ & [a], [\neg a], [a; b \leftarrow a], [c \leftarrow not\ \ a], [d \leftarrow not\ \ a, not\ \ e],\\ & [e \leftarrow not\ \ f], [e \leftarrow not\ \ g], [f], [g], [g \leftarrow not\ \ c], [h \leftarrow not\ \ g],\\ & [i \leftarrow not\ \ j], [j \leftarrow not\ \ i]\ \ \}\end{aligned}$$

i.e. $\{A_a^s, A_{\neg a}^s, A_b^s, A_c^s, A_d^s, A_e^s, A_e'^s, A_f^s, A_g^s, A_g'^s, A_h^s, A_i^s, A_j^s\}$. *The set of w-arguments of P is*

$$\begin{aligned}\mathcal{A}rgs^w(P) = \{\ & [a \leftarrow not\ \ \neg a, not\ \ \bot], [\neg a \leftarrow not\ \ a, not\ \ \bot],\\ & [a \leftarrow not\ \ \neg a, not\ \ \bot; b \leftarrow a, not\ \ \neg b, not\ \ \bot],\\ & [c \leftarrow not\ \ a, not\ \ \neg c, not\ \ \bot], [d \leftarrow not\ \ a, not\ \ e, not\ \ \neg d, not\ \ \bot],\\ & [e \leftarrow not\ \ f, not\ \ \neg e, not\ \ \bot], [e \leftarrow not\ \ g, not\ \ \neg e, not\ \ \bot],\\ & [f \leftarrow not\ \ \neg f, not\ \ \bot], [g \leftarrow not\ \ \neg g, not\ \ \bot],\\ & [g \leftarrow not\ \ c, not\ \ \neg g, not\ \ \bot], [h \leftarrow not\ \ g, not\ \ \neg h, not\ \ \bot],\\ & [i \leftarrow not\ \ j, not\ \ \neg i, not\ \ \bot], [j \leftarrow not\ \ i, not\ \ \neg j, not\ \ \bot]\ \ \}\end{aligned}$$

i.e. $\{A^w_a, A^w_{\neg a}, A^w_b, A^w_c, A^w_d, A^w_e, A'^w_e, A^w_f, A^w_g, A'^w_g, A^w_h, A^w_i, A^w_j\}$. *Finally,*

$$\mathcal{A}rgs(P) = \mathcal{A}rgs^s(P) \cup \mathcal{A}rgs^w(P)$$

At this point we have defined how arguments are built. We now move on to defining the attacking relation between these arguments. Instead of [PS97]'s definition of attack, with undercut and rebut (see Def. 25), our definition is as follows. If an argument for an objective literal L (denoted by A_L) has a default negation *not* L' in it, any argument for L' attacks (by undercut) A_L. The other attacking relation (named rebut) states that an argument also attacks another one when both arguments have complementary conclusions (i.e. one concludes L and the other $\neg L$). With strong and weak arguments, *rebut can be reduced to undercut*. So, we can say informally that "an argument for a conclusion L *attacks* an argument with an assumption *not* L". Such a "notion of attack" shows that we need to make both the conclusions and the assumptions of an argument precise before defining an attack.

Definition 39 (Conclusions and Assumptions) *Let A_L be an argument for L. The* conclusions *of A_L, $Conc(A_L)$, is the set of all objective literals that appear in the head of rules in A_L. The* assumptions *of A_L, $Assump(A_L)$, is the set of all default literals appearing in the bodies of rules in A_L.*

Example 14 *Assume the R^s_P and R^w_P of Example 10. The strong and the weak argument for b are, respectively, $A^s_b = [a; b \leftarrow a]$ and $A^w_b = [a \leftarrow not\ \neg a, not\ \bot; b \leftarrow a, not\ \neg b, not\ \bot]$. The conclusions and assumptions for both arguments are as follows*

$$Conc(A^s_b) = Conc(A^w_b) = \{a, b\}$$
$$Assump(A^s_b) = \emptyset$$
$$Assump(A^w_b) = not\ \{\neg a, \neg b, \bot\}$$

Intuitively, both strong and weak arguments can be attacked in the same way. Since a (weak or strong) argument may make assumptions, other arguments for the complement of one such assumption may attack it. In other words, an argument with *not* L can be attacked by arguments for L. This definition of attack encompasses two cases:

- arguments that are directly conflicting, e.g. an argument for L (with *not* $\neg L$) can be attacked by an argument for $\neg L$, and
- any weak argument A^w_L (and also a strong argument A^s_L which verifies *not* $\bot \in Assump(A^s_L)$) can be attacked by every argument for $\bot$.

However, it does not make sense to attack arguments for objective literals if they do not lead to *falsity*. By "an objective literal L leads to *falsity*" we mean that there is an argument A_L such that $A_\bot$ is built based on such an argument, e.g. $A_\bot : A_L + [\bot \leftarrow L, not\ L']$[4] . However, this proposal considers only the objective literals that are in the body of the rule for $\bot$ (called $r_\bot$) because these literals immediately lead to *falsity*. We assume the involvement of other objective literals is not so strong as those in the body of $r_\bot$. To clarify the above description, see Example 15. (We further assume they can be detected in a process of "belief revision", e.g. [DP97, DPS97]. A full discussion of this issue is beyond the scope of this proposal.) We define objective literals *directly conflicting with* $A_\bot$ as follows:

Definition 40 (Directly Conflict with $A_\bot$) *Let $A_\bot$ be an argument for $\bot$, '$\bot \leftarrow$ Body' be the rule for $\bot$ in $A_\bot$, and $\{L_1, \ldots, L_n\}$ be the set of all objective literals in Body. The set of objective literals* directly conflicting with $A_\bot$ *is*

$$DC(A_\bot) = \{\bot\} \cup \{L_1, \ldots, L_n\}.$$

Definition 41 (Attack) *Let P be an ELPd. An argument A_L of P for L* attacks *an argument $A_{L'}$ of P for L' iff*

- *L is the symbol $\bot$, not $\bot \in Assump(A_{L'})$, and $L' \in DC(A_L)$; or*
- *L is an objective literal different from $\bot$, and not $L \in Assump(A_{L'})$.*

Example 15 *Consider the program $\{a;\ b \leftarrow c;\ c;\ d; e \leftarrow not\ \bot;\ \bot \leftarrow a, b\}$. The strong argument for $\bot$ is $A^s_\bot = [c; b \leftarrow c; a; \bot \leftarrow a, b]$ and so $DC(A^s_\bot) = \{\bot, a, b\}$. Although a rule for c belongs to the sequence of $A^s_\bot$, its involvement is not so strong as a and b. Furthermore, d does not lead to* falsity. *Despite of the fact that 'not $\bot$' is in the sequence of A_e, the literal $e \notin DC(A^s_\bot)$. Thus, $A^s_\bot$ can attack both A^w_a and A^w_b, but may not attack A^w_c, A^w_d, A^s_e, and A^w_e.*

Based on the simplifications that we have performed in Prakken's definitions, we can further say that the evaluation of a strong argument for an objective literal L does not consider the existence of arguments for $\neg L$, nor the presence of *falsity*. However, a weak argument for L is evaluated by looking at both of them. Moreover, if a strong argument is attacked, then the weaker version of it is also attacked. Given that $\mathcal{A}rgs(P)$ contains strong arguments for a given objective literal and also contains the weak corresponding arguments for it (cf. Def. 38), the following holds:

[4]'A+B' means to concatenate the arguments 'A' and 'B' in terms of well-defined sequences. Since the operator '+' is only refereed in examples, we do not define it yet. The next chapter has a formal definition of such an operator (see Def. 58).

Proposition 12 *Let A^s_L be an argument, and A^w_L be its weak corresponding argument. If a (strong or weak) argument $A_{L'}$ attacks A^s_L, then $A_{L'}$ also attacks A^w_L.*

Proof. Assume, by contradiction, that there is an argument $A_{L'}$ that attacks A^s_L and does not attack A^w_L. If $A_{L'}$ attacks A^s_L then there exists a *not* $L' \in Assump(A^s_L)$. Given that $Assump(A^s_L) \subseteq Assump(A^w_L)$ then there also exists *not* $L' \in Assump(A^w_L)$. So $A_{L'}$ also attacks A^w_L, and we have a contradiction ■

Note that the converse is not necessarily true. I.e. it may happen that an argument attacks A^w_L and does not attack A^s_L.

Since attacking arguments can in turn be attacked by other arguments, comparing arguments is not enough to determine their acceptability w.r.t. the set of overall arguments. What is also required is a definition that determines the acceptable arguments on the basis of all the ways in which they interact. In other words, the acceptability of arguments is obtained through the interaction of arguments, by proposing arguments and opposing them. A subset S of proposed arguments of P is acceptable only if the set of arguments of P, $\mathcal{A}rgs(P)$, does not have some valid opposing argument attacking the proposed arguments in S. As in [Dun93, PS97], we demand acceptable sets to contain all such arguments. Two questions remain open: how to obtain opposing arguments and, among these, which are valid?

An opposing argument for a proposed argument which makes an assumption, say *not* L, is simply an argument for conclusion L in $\mathcal{A}rgs(P)$. For an opposing argument A^o to be valid for attacking a proposed argument A^p in S, S should not have another argument that, in turn, attacks A^o (i.e. another argument that reinstates[5] A^p). In this case, we say that S cannot defend itself against A^o. This motivation points to a definition of acceptable sets of arguments in P such as a set S is *acceptable* if it can attack all opposing arguments from $\mathcal{A}rgs(P)$. So, we can say that

> A proposed argument A^p is acceptable w.r.t. a set S of acceptable arguments if and only if each opposing argument A^o attacking A^p is (counter-)attacked by an argument in S.

This notion of acceptable argument has been introduced without considering two of [PS97]'s definitions, viz. *defeat* and *strictly defeat*, because both of them are based on definitions of *undercut* and *rebut*. However, it is still necessary to determine how strong arguments and weak arguments should interact w.r.t. such a set S of arguments. Based on the idea of reinstatement, both attacked and counter-attacking arguments should be of the same kind. For instance, if a proposing

[5]For details, see Example 1.

argument is strong (resp. weak) then every counter-attack against its opposing argument should be strong (resp. weak). A similar reasoning can be applied to opposing arguments. Therefore, proposed (resp. opposing) arguments should be of the same kind.

Remark 13 *In the remainder of this dissertation we will use the notation p and o to distinguish the proposed argument from the opponent one, i.e. p (resp. o) is a (strong or weak) proposed (resp. opponent) argument.*

In what concerns the kind of arguments, there are four possibilities of interaction between a proposed argument A^p and an opposing argument A^o as represented in Table 3.1. The first mode of interaction considers a set of weak arguments only. The second mode considers a set with strong arguments. The third/fourth consider a set with both strong and weak arguments. (In the following, these interactions are discussed in detail.) Then the definition of arguments' acceptability (and the corresponding characteristic function) is generalized by parameterizing with the possible kinds of arguments, viz. strong arguments and weak arguments.

$A^p \backslash A^o$	s	w
s	s, s	s, w
w	w, s	w, w

Table 3.1: Ways of interacting arguments

Definition 42 (Acceptable Argument) *Let P be an ELPd, p (resp. o) be the kind (strong or weak) of the proposed (resp. opposing) argument of P, and $S \subseteq \mathcal{A}rgs^p(P)$.*

An argument $A_L \in \mathcal{A}rgs^p(P)$ is an acceptable$_{p,o}$ argument *w.r.t. S iff each argument $A_{L'} \in \mathcal{A}rgs^o(P)$ attacking A_L is attacked by an argument $A_{L''} \in S$.*

Example 16 *Following Example 13, Figure 3.3 illustrates the attacking relation between proposed strong arguments and opposing weak arguments (see Remark 14 to understand better the notation used in the figure).* Acceptable$_{s,w}$ *arguments w.r.t. $\mathcal{A}rgs(P)$ are A^s_b, A^s_d, A^s_c, A^s_g, A'^s_g, A^s_f, $A^s_{\neg a}$ and A^s_a.*

Remark 14 (Notation used in the Figures) *Let A and B be arguments of an ELPd $\mathcal{P}$. Arguments are represented as nodes. A solid line from A to B means "A attacks B", a dotted line from A to B means "A is built based on B", and a line with dashes means "A reinstates B". A round node means "it is an acceptable argument" and a square node means "it is not an acceptable argument", which are w.r.t. the set of arguments of $\mathcal{P}$.*

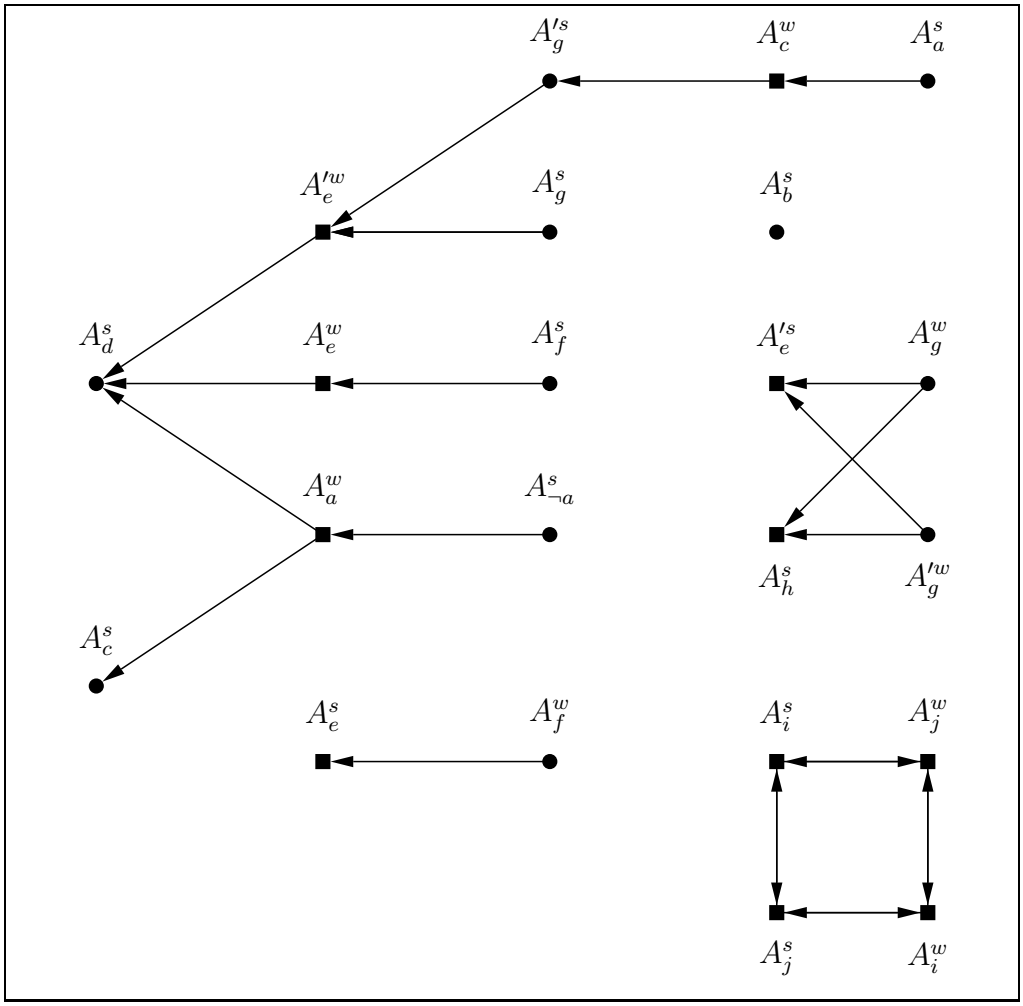


Figure 3.3: Proponent strong arguments and opposing weak arguments of Example 13

[PS97]'s proposal is in accordance with the 'Compositional Principle' of [Vre97]: "If an argument $A_{L'}$ is a sub-argument of argument A_L, and $A_{L'}$ is not acceptable w.r.t. a set of arguments S, then A_L is also not acceptable w.r.t. S". Furthermore, [PS97] imposes the rule that an argument and its sub-arguments have the same rules in their sequence of rules (cf. Def. 24). However, we could define sub-arguments based on definitions of both conclusions and assumptions. Nevertheless, [Vre97]'s principle is respected by our proposal, and the following holds:

Proposition 15 *Assume two arguments A^p_L and $A^p_{L'}$ such that*

$$Conc(A^p_{L'}) \subseteq Conc(A^p_L) \text{ and } Assump(A^p_{L'}) \subseteq Assump(A^p_L)$$

If $A^p_{L'}$ is not acceptable$_{p,o}$ w.r.t. a set of arguments S then A^p_L is also not acceptable$_{p,o}$ w.r.t. S.

Proof. Assume, by contradiction, that there is an argument $A^o_{L''}$ in S that attacks $A^p_{L'}$ and does not attack A^p_L. If $A^o_{L''}$ attacks $A^p_{L'}$ then *not* $L'' \in Assump(A^p_{L'})$. Given that $Assump(A^p_{L'}) \subseteq Assump(A^p_L)$ then *not* $L'' \in Assump(A^p_L)$. So $A^o_{L''}$ also attacks A^p_L. Contradiction ■

The acceptability of an argument is local to Ag, since this argument depends on the set P of rules of Ag from which the argument is built. Even when handling the presence of *falsity* in Ag (i.e. $A_\perp$) or contradictory arguments (e.g. A_L and $A_{\neg L}$), these arguments are always built over P. Similarly to [Dun93, Dun95], we formalize the concept of acceptable arguments with a fixpoint operator. The intuitive idea of the characteristic function is that the set of acceptable arguments is constructed step-by-step. First, an empty set is assumed to be the initial set of acceptable arguments. Then, all proposed arguments which are acceptable w.r.t. to the empty set (i.e. which are not attacked by any opposing argument) are collected into a set, S^1. After that, all proposed arguments that are acceptable w.r.t. arguments in S^1 are added in S^2. More precisely, each argument that has attacking opposing arguments is added if all those opposing arguments are attacked by an argument already in S^i. The resulting set is S^{i+1}. This step is repeated until a set S^λ is obtained to which no new proposed argument can be added.

Definition 43 (Characteristic Function) *Let P be an ELPd, and p (resp. o) be the kind (strong or weak) of the proposed (resp. opposing) argument of P, and $S \subseteq \mathcal{A}rgs^p(P)$. The* characteristic function *p o of P and over S is:*

$$F^{p,o}_P : 2^{\mathcal{A}rgs(P)} \rightarrow 2^{\mathcal{A}rgs(P)}$$
$$F^{p,o}_P(S) = \{Arg \in \mathcal{A}rgs(P) \mid Arg \text{ is } acceptable_{p,o} \text{ w.r.t. } S\}$$

Furthermore, if an argument A_L^p is $acceptable_{p,o}$ w.r.t. S, then A_L^p is also $acceptable_{p,o}$ w.r.t. any superset of S. In fact:

Proposition 16 *$F_P^{p,o}$ is monotonic.*

Proof. Assume that S^1 and S^2 are subsets of $\mathcal{A}rgs(P)$. We have to prove that $\forall S^1, S^2$: if $S^1 \subseteq S^2$ then $F_P^{p,o}(S^1) \subseteq F_P^{p,o}(S^2)$. I.e. if, assuming $S^1 \subseteq S^2$, an argument $A_L \in F_P^{p,o}(S^1)$ then $A_L \in F_P^{p,o}(S^2)$. If A_L is $acceptable_{p,o}$ w.r.t. S^1 then (i) A_L has no argument in $\mathcal{A}rgs(P)$ that attacks it; or (ii) there is an argument $A_{L'} \in \mathcal{A}rgs(P)$ that attacks A_L but there is an argument $A_{L''}$ in S^1 that attacks $A_{L'}$. If A_L is $acceptable_{p,o}$ w.r.t. S^1 because of (i), then A_L is also $acceptable_{p,o}$ w.r.t. S^2 because there is still no argument in $\mathcal{A}rgs(P)$ that attacks A_L. If A_L is $acceptable_{p,o}$ w.r.t. S^1 because of (ii), then $A_{L''} \in S^2$, since $S^1 \subseteq S^2$, and so A_L is also $acceptable_{p,o}$ w.r.t. S^2 ■

Being monotonic, it is guaranteed that $F_P^{p,o}$ always has a least fixpoint (according to the set inclusion ordering over sets of arguments):

Proposition 17 *Define for any P the following sequence of sets of arguments:*

- $S^0 = \emptyset$
- $S^{i+1} = F_P^{p,o}(S^i)$

Given that $F_P^{p,o}$ is monotonic, there must exist a smallest λ such that S^λ is a fixpoint of $F_P^{p,o}$, and $S^\lambda = lfp(F_P^{p,o})$.

Proof. The result follows immediately from the monotonicity of $F_P^{p,o}$, given the well-known Knaster-Tarski Theorem [Tar55] ■

The following example shows that $lfp(F_P^{p,o})$ is well-behaved, i.e. arguments in it are $acceptable_{p,o}$ w.r.t. the set of all arguments of P. By definition $lfp(F_P^{p,o})$ is minimal, which guarantees that it does not contain any argument of which acceptance is not required.

Example 17 *Assume the $\mathcal{A}rgs(P)$ presented in Example 13. The iterative construction of the set of $acceptable_{s,w}$ arguments w.r.t. $\mathcal{A}rgs(P)$ is as follows:*

- $S^0 = \emptyset$
- $S^1 = F_P^{s,w}(S^0) = \{A_g^s, A_f^s, A_a^s, A_{\neg a}^s, A_b^s\}$
- $S^2 = F_P^{s,w}(S^1) = \{A_g^s, A_f^s, A_a^s, A_{\neg a}^s, A_b^s, A_d^s, A_c^s, A_g'^s\}$

- $S^3 = F_P^{s,w}(S^2) = S^2 = lfp(F_P^{s,w})$

Similarly to [PS97], we formalize the status of an argument. By knowing the set of all acceptable$_{p,o}$ arguments of P, the arguments from $\mathcal{A}rgs(P)$ are split into three classes, viz justified$_P^{p,o}$, overruled$_P^{p,o}$ and defensible$_P^{p,o}$. Nevertheless, our definition of overruled is different from [PS97]'s proposal. In their proposal, the restriction applies that overruled arguments cannot be also justified (see Def. 30) and so [PS97]'s argumentation semantics is always consistent. Since we aim to obtain a paraconsistent way of reasoning, the status of an argument is defined as follows:

Definition 44 (Justified, Overruled or Defensible Argument) *Let P be an ELPd, p (resp. o) be the kind (strong or weak) of an argument of P, and $F_P^{p,o}$ be the characteristic function p o of P and over $\mathcal{A}rgs(P)$. An argument A_L^p is*

- justified$_P^{p,o}$ *iff A_L^p is in $lfp(F_P^{p,o})$*
- overruled$_P^{p,o}$ *iff the A_L^o corresponding to it is attacked by a justified$_P^{p,o}$ argument*
- defensible$_P^{p,o}$ *iff it is neither a justified$_P^{p,o}$ nor an overruled$_P^{p,o}$ argument.*

Remark 18 ($JustArgs_P^{p,o}$) *We denote the $lfp(F_P^{p,o})$ by the* set of justified$_P^{p,o}$ arguments*, $JustArgs_P^{p,o}$.*

We may deduce overruled arguments of $\mathcal{A}rgs(P)$ based on the greatest fixpoint of the characteristic function. We obtain the greatest fixpoint in a similar way to the least fixpoint, i.e. constructed step-by-step. However, $\mathcal{A}rgs(P)$ is assumed to be the initial set of arguments. Then, all proposed arguments which are acceptable w.r.t. $\mathcal{A}rgs(P)$ are collected into a set, S^1. After that, all proposed arguments that are acceptable w.r.t. S^1 are added in S^2. In other words, every proposed argument which is attacked by opposing arguments in S^i and is not reinstated by arguments in S^i, is not added into set S^{i+1}. This step is repeated until a set S^λ is obtained to which no proposed argument can be deduced, i.e every attacked argument in S^λ is reinstated by an argument in S^λ. Intuitively, we can assume that all acceptable arguments in S^λ are either justified or defensible arguments. Thus, every argument which does not belong to S^λ is an overruled argument.

Since $F_P^{p,o}$ always has a greatest fixpoint (according to the set inclusion ordering over sets of arguments), the following holds:

Proposition 19 *Define for any P the following sequence of sets of arguments:*

- $S^0 = \mathcal{A}rgs^p(P)$
- $S^{i+1} = F_P^{p,o}(S^i)$

1. *Given that $F_P^{p,o}$ is monotonic, there must exist a smallest λ such that S^λ is a fixpoint of $F_P^{p,o}$, and $S^\lambda = gfp(F_P^{p,o})$.*

2. *Moreover, $gfp(F_P^{p,o}) = F^{p,o\downarrow\omega}$.*

Proof. The result follows immediately from the monotonicity of $F_P^{p,o}$, again given the well-known Knaster-Tarski Theorem [Tar55] ■

Then we relate both greatest fixpoint and least fixpoint as follows. The $gfp(F_P^{o,p})$ contains both justified$_P^{o,p}$ and defensible$_P^{o,p}$ arguments w.r.t. $\mathcal{A}rgs(P)$, and the $lfp(F_P^{p,o})$ contains the justified$_P^{p,o}$ arguments w.r.t. $\mathcal{A}rgs(P)$.

Lemma 20 $gfp(F_P^{o,p}) = \{A_{L_1}^o : \neg(\exists A_{L_2}^p \in lfp(F_P^{p,o}) \mid A_{L_2}^p \text{ attacks } A_{L_1}^o)\}$.

Proof. Let $S = \{A_{L_1}^o : \neg(\exists A_{L_2}^p \in lfp(F_P^{p,o}) \mid A_{L_2}^p \text{ attacks } A_{L_1}^o)\}$. We prove that (1) S is a fixpoint of $F_P^{o,p}$, and that (2) S is the greatest fixpoint of $F_P^{o,p}$.

1. $F_P^{o,p}(S) = S$. By definition, $A_{L_1}^o$ is acceptable$_{o,p}$ w.r.t. S iff it is not attacked or it is attacked by $A_{L_2}^p$ and $\exists A_{L_3}^o \in S$ attacking $A_{L_2}^p$. So, $A_{L_1}^o$ is acceptable$_{o,p}$ w.r.t. S if for $\forall A_{L'}^p$ that attacks $A_{L_1}^o$ then $\exists A_{L''}^o \in S$ attacking $A_{L'}^p$. By definition of $A_{L_1}^o$, if $A_{L'}^p$ attacks $A_{L_1}^o$ then $A_{L'}^p \notin lfp(F_P^{p,o})$. Therefore, $A_{L'}^p$ is attacked by $A_{L''}^o$ that is not attacked by any argument in $lfp(F_P^{p,o})$. But in this case, by definition of S, $A_{L''}^o \in S$. Thus, $A_{L_1}^o$ is acceptable$_{o,p}$ w.r.t. S. We have proven that $S \subseteq F_P^{o,p}(S)$. Now, we have to prove that $S \supseteq F_P^{o,p}(S)$. Let $A_L^o \in F_P^{o,p}(S)$. We must show that $\neg(\exists A_{L_2}^p \in lfp(F_P^{p,o}) \mid A_{L_2}^p \text{ attacks } A_L^o)$. Assume, by contradiction, $\exists A_{L_2}^p lfp(F_P^{p,o})$ such that $A_{L_2}^p$ attacks A_L^o. By hypothesis, A_L^o is acceptable$_{o,p}$ w.r.t. S so there exists an $A_{L_3}^o \in S$ attacking $A_{L_2}^p$. Given that $A_{L_3}^o \in S$, $\neg\exists A_{L_4}^p \in lfp(F_P^{p,o})$ attacking $A_{L_3}^o$. So, $A_{L_2}^p \notin lfp(F_P^{p,o})$ *q.e.d.*

2. If $S' = F_P^{o,p}(S')$ then $S' \subseteq S$. We have to prove that $\forall A_L^o \in S' : \neg(\exists A_{L_2}^p \in lfp(F_P^{p,o}) \mid A_{L_2}^p \text{ attacks } A_L^o)$, assuming that $S' = F_P^{o,p}(S')$. In other words, we have to prove that, for every $A_L^o \in S'$, every argument $A_{L_2}^p$ attacking A_L^o does not belong to $lfp(F_P^{p,o})$. Since $lfp(F_P^{p,o})$ can be obtained by the iteration of $F_P^{p,o}$, we prove this by induction on the iteration. More precisely, let $A_L^o \in S'$ and $C = \{A_{L'}^p \mid A_{L'}^p \text{ attacks } A_L^o\}$. We prove by induction that $\forall_i F_P^{p,o\uparrow i} \cap C = \emptyset$.

   Base: Trivial, since $\emptyset \cap C = \emptyset$.

   Induction: Assume that $F_P^{p,o\uparrow i} \cap C = \emptyset$. $F_P^{p,o\uparrow i+1} = \{A_{L'}^p : if A_{L''}^o \text{ attacks } A_{L'}^p \text{ then } \exists A_{L'''}^p \in F_P^{p,o\uparrow i} \text{ that attacks } A_{L''}^o\}$. We have to prove that any such $A_{L'}^p \notin C$, i.e. $A_{L'}^p$ does not attack any $A_L^o \in S'$. If $A_{L'}^p$ attacks some $A_L^o \in S'$

then there exists an $A^o_{L''} \in S'$ attacking $A^p_{L'}$. But, in this case, there does not exist $A^p_{L'''} \in F_P^{p,o\uparrow i}$ attacking $A^o_{L''}$ because, by induction hypothesis, $F_P^{p,o\uparrow i} \cap C = \emptyset$. ■

Lemma 21 $lfp(F_P^{p,o}) = \{A^p_{L_1} \ : \ \neg(\exists A^o_{L_2} \in gfp(F_P^{o,p}) \mid A^o_{L_2} \text{ attacks } A^p_{L_1})\}$.

Proof. Let $S = \{A^p_{L_1} \ : \ \neg(\exists A^o_{L_2} \in gfp(F_P^{o,p}) \mid A^o_{L_2} \text{ attacks } A^p_{L_1})\}$. We prove that (1) S is a fixpoint of $F_P^{p,o}$, and that (2) S is the least fixpoint of $F_P^{p,o}$.

1. $F_P^{p,o}(S) = S$. By definition, $A^p_{L_1}$ is acceptable$_{p,o}$ w.r.t. S iff it is attacked by $A^o_{L_2}$ and $\exists A^p_{L_3} \in S$ attacking $A^o_{L_2}$. So, $A^p_{L_1}$ is acceptable$_{p,o}$ w.r.t. S if for $\forall A^o_{L'}$ that attacks $A^p_{L_1}$ then $\exists A^p_{L''} \in S$ attacking $A^o_{L'}$. By definition of $A^p_{L_1}$, $A^o_{L'} \notin gfp(F_P^{o,p})$. So, if $A^o_{L'}$ attacks $A^p_{L_1}$ then $A^o_{L'} \notin gfp(F_P^{o,p})$. Therefore, $A^o_{L'}$ is attacked by $A^p_{L''}$ that is not attacked by any argument in $gfp(F_P^{o,p})$. But in this case, by definition of S, $A^p_{L''} \in S$. Thus, $A^p_{L_1}$ is acceptable$_{p,o}$ w.r.t. S. We have proven that $S \subseteq F_P^{p,o}(S)$. Now, we have to prove that $S \supseteq F_P^{p,o}(S)$. Let $A^p_L \in F_P^{p,o}(S)$. We should also prove that $\neg(\exists A^o_{L_2} \in gfp(F_P^{o,p}) \mid A^o_{L_2}$ attacks $A^p_L)$. Assume, by contradiction, $A^o_{L_2} \in gfp(F_P^{o,p})$ and $A^o_{L_2}$ attacks A^p_L. By induction hypothesis, if A^p_L is acceptable$_{p,o}$ w.r.t. S then there exists an $A^p_{L_3} \in S$ attacking $A^o_{L_2}$. Given that $A^p_{L_3} \in S$, $\neg\exists A^o_{L_4} \in gfp(F_P^{o,p})$ attacking $A^p_{L_3}$. So, $A^o_{L_2} \notin gfp(F_P^{o,p})$ *q.e.d.*

2. If $S' = F_P^{p,o}(S')$ then $S \subseteq S'$. We have to prove that $\forall A^p_L \notin S' : \exists A^o_{L_2} \in gfp(F_P^{o,p}) \mid A^o_{L_2}$ attacks A^p_L, assuming that $S' = F_P^{p,o}(S')$. In other words, we have to prove that for every $A^p_L \notin S'$ there exists an argument $A^o_{L_2}$ attacking A^p_L belonging to $gfp(F_P^{o,p})$. Since $gfp(F_P^{o,p})$ can be obtained by the iteration of $F_P^{o,p}$, we prove this by induction on the iteration. More precisely, let $C = \{A^o_{L'} \mid A^o_{L'}$ attacks $A^p_L\}$. We prove by induction that $\forall_i F_P^{o,p\downarrow i} \cap C \neq \emptyset$.

   Base: Clearly, $\mathcal{A}rgs(P) \cap C = C$. Since S' is a fixpoint of $F_P^{p,o}$ and $A^p_L \notin S'$ then there must exist some argument attacking A^p_L, i.e. C cannot be empty.

   Induction: Assuming that $F_P^{o,p\downarrow i} \cap C \neq \emptyset$. $F_P^{o,p\downarrow i+1} \cap C = \{A^o_{L'} :$ if $A^p_{L''}$ attacks $A^o_{L'}$ then $\exists A^o_{L'''} \in F_P^{o,p\downarrow i}$ that attacks $A^p_{L''}\}$. We have to prove that $A^o_{L_1} \in C$, i.e. $\exists A^o_{L_1}$ attacking A^p_L. If $\forall_i$ $(\exists A^o_{L_1} \in F_P^{o,p\downarrow i+1} \mid A^o_{L_1}$ attacks $A^p_L)$ then $(\exists A^o_{L_3} \in F_P^{o,p\downarrow i} \mid A^o_{L_3}$ attacks $A^p_{L_2})$ because, by induction hypothesis, $F_P^{o,p\downarrow i} \cap C \neq \emptyset$ ■

Then, the following holds:

Theorem 22 *A^p_L is overruled$^{p,o}_P$ iff the corresponding A^o_L is not in $gfp(F_P^{o,p})$.*

Proof. If $A^p_{L_1}$ is $overruled^{p,o}_P$ then there exists an $A^p_{L_2} \in lfp(F^{p,o}_P)$ attacking the corresponding $A^o_{L_1}$ to $A^p_{L_1}$. Based on Lemma 20, $A^o_{L_1} \notin gfp(F^{o,p}_P)$. If $A^o_{L_1} \notin gfp(F^{o,p}_P)$ then, based on Lemma 21, there exists an $A^p_{L_2} \in lfp(F^{p,o}_P)$ attacking $A^o_{L_1}$. So, by Definition 44, $A^p_{L_1}$ is $overruled^{p,o}_P$ ■

Example 18 *Assume* $\mathcal{A}rgs^s(P) = \{A^s_a, A^s_{\neg a}, A^s_b, A^s_c, A^s_d, A^s_e, A'^s_e, A^s_f, A^s_g, A'^s_g, A^s_h, A^s_i, A^s_j\}$ *from Example 13, and* $lfp(F^{s,w}_P) = \{A^s_a, A^s_{\neg a}, A^s_b, A^s_c, A^s_d, A'^s_g, A^s_g, A^s_f\}$ *from Example 17. Figure 3.4 illustrates the attacking relation between proposed weak arguments and opposing strong ones (see Remark 14 to understand better the notation used in the figure). Then, the* $gfp(F^{w,s}_P)$ *is obtained as follows:*

- $S^0 = \mathcal{A}rgs^w(P)$
- $S^1 = F^{w,s}_P(S^0) = \{A^w_g, A'^w_g, A^w_f, A^w_i, A^w_j\}$
- $S^2 = F^{w,s}_P(S^1) = \{A^w_g, A^w_f, A^w_i, A^w_j\}$
- $S^3 = F^{w,s}_P(S^2) = S^2 = gfp(F^{w,s}_P)$

Both A^s_g *and* A^s_f *are* $justified^{s,w}_P$ *arguments, and both* A^s_i *and* A^s_j *are* $defensible^{s,w}_P$ *arguments. The* $overruled^{s,w}_P$ *arguments are* A^s_e, A'^s_e *and* A^s_h. *The other arguments in* $\mathcal{A}rgs(P)$ *are both* $justified^{s,w}_P$ *and* $overruled^{s,w}_P$, *viz.* A^s_a, $A^s_{\neg a}$, A^s_b, A^s_c, A^s_d *and* A'^s_g.

Not all sets of justified arguments have the same "level of acceptability". It depends on the kind of both proposed and opposing arguments. Most of these sets do not have a justified argument attacking another justified one, i.e. they do not accept conflicts between justified arguments. On the other hand, some of them might have contradictory arguments, i.e. they may have an argument for L and another for $\neg L$. So, we first define what we mean by *conflict-free* and by *contradictory* sets of arguments. Then establish the relation between justified arguments and a contradiction: every justified argument is either contradictory, based on contradiction, or non contradictory w.r.t. $\mathcal{A}rgs(P)$.

Definition 45 (Conflict-free Set of Arguments) *A set of arguments* S *is* conflict-free *iff there is no argument in* S *that attacks an argument in* S.

Definition 46 (Contradictory Set of Arguments) *A set of arguments* S *is contradictory w.r.t.* L *iff*

- L *is the symbol* $\bot$, *and there exist an argument for* falsity *in* S; *or*
- L *is an objective literal different from* $\bot$, *and there exists both an argument for* L *and an argument for* $\neg L$ *in* S.

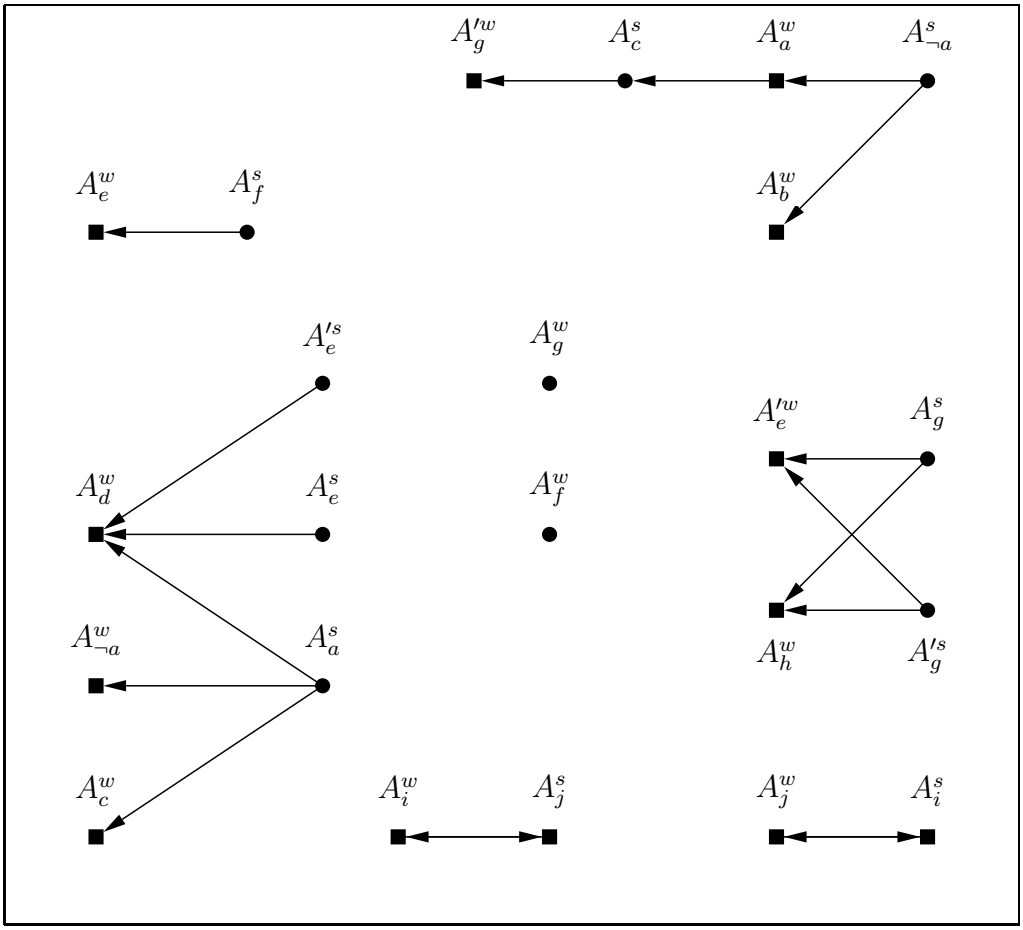


Figure 3.4: Proposed weak arguments and opposing strong arguments of Example 13

We say that S is non contradictory *iff there is no literal L in S such that S is contradictory w.r.t. L.*

We define three sub-statuses for relating justified arguments with contradiction — contradictory, based-on-contradiction, or non-contradictory — as follows:

- Intuitively, justified arguments are *contradictory* if they are related to a contradictory set of arguments, i.e. if a set S of justified arguments is contradictory w.r.t. an objective literal L, then every justified argument for such L is contradictory w.r.t. S.

  Another condition is based on objective literals directly conflicting with $A^p_\perp$, which means that if $A^p_\perp$ is a justified argument then, by the first condition, it is also contradictory. Thus, every justified argument for an objective literal in $DC(A^p_\perp)$ is contradictory because it immediately causes *falsity*.

- An argument is *based on contradiction* if one of the following conditions is satisfied: (i) A^p_L is reinstated by a contradictory argument or a based-on-contradiction argument, or (ii) A^p_L is built based on a contradictory argument.

- An argument is *non contradictory* otherwise, i.e. if it is not involved at all with any contradiction.

We say that an argument that reinstates another argument is its *counter-attack*:

Definition 47 (Counter-Attack) *Let P be an ELPd, $S \subseteq \mathcal{A}rgs(P)$, A_L be an argument in S, and $A_{L'}$ be an argument in $\mathcal{A}rgs(P)$ attacking A_L. A* counter-attack *for A_L against $A_{L'}$ is an argument in S that attacks $A_{L'}$. $CA_{A_L}(A_{L'}, S)$ is the set of all counter-attacks for A_L against $A_{L'}$ in S.*

Definition 48 (Relation to Contradiction) *Let P be an ELPd. A justified$^{p,o}_P$ argument A^p_L is*

- contradictory$^{p,o}_P$ *iff*
  - *$JustArgs^{p,o}_P$ is contradictory w.r.t. L, or*
  - *there exists an $^{p,o}_P$ argument $A^p_\perp$ in $Justargs^{p,o}_P$ and $L \in DC(A^p_\perp)$;*
- based-on-contradiction$^{p,o}_P$ *iff*
  - *for all $A^o_{L'}$ attacking A^p_L there exists a contradictory$^{p,o}_P$ or based-on-contradiction$^{p,o}_P$ argument in $CA_{A_L}(A^o_{L'}, JustArgs^{p,o}_P)$, or*

– *there exists an* $L' \in Conc(A_L^p)$*, different from* L *and* $\bot$*, such that* $JustArgs_P^{p,o}$ *is contradictory w.r.t.* L'*;*

- non-contradictory$_P^{p,o}$ *iff*

  it is neither contradictory$_P^{p,o}$ *nor based-on-contradiction*$_P^{p,o}$*.*

Proposition 23 *Let* L *be an objective literal different from* $\bot$*. A justified*$_P^{p,o}$ A_L^p *argument is non-contradictory*$_P^{p,o}$ *if and only if there does not exist an* $L' \in Conc(A_L^p)$ *such that* $JustArgs_P^{p,o}$ *is contradictory w.r.t.* L'*, and every counter-attack for* A_L^p *is a non-contradictory*$_P^{p,o}$ *argument.*

Proof. We have to prove that A_L^p is non-contradictory$_P^{p,o}$ iff (1) $\neg\exists L' \in Conc(A_L^p)$ such that $JustArgs_P^{p,o}$ is contradictory w.r.t. L', and (2) $\forall A_{L'}^o$ attacking A_L^p, every $A_{L''}^p \in CA_{A_L}(A_{L'}^o, JustArgs_P^{p,o})$ is a non-contradictory$_P^{p,o}$ argument.

1. We assume, by absurdum, that A_L^p is non-contradictory$_P^{p,o}$ and $\exists L' \in Conc(A_L^p)$ such that $JustArgs_P^{p,o}$ is contradictory w.r.t. L'. If $L = L'$ then, by definition, A_L^p is contradictory$_P^{p,o}$; otherwise, A_L^p is based-on-contradiction$_L^{p,o}$. Absurdum.

2. We assume, by absurdum, that A_L^p is non-contradictory$_P^{p,o}$ and $\exists A_{L'}^o$ attacking $A_L^p \mid \exists A_{L''}^p \in CA_{A_L}(A_{L'}^o, JustArgs_P^{p,o})$ such that $A_{L''}^p$ is not a non-contradictory$_P^{p,o}$ argument. If $A_{L''}^p$ is either a contradictory$_P^{p,o}$ or a based-on-contradiction$_P^{p,o}$ argument then, by definition, A_L^p is based-on-contradiction$_P^{p,o}$. Absurdum ■

Example 19 *Following Example 18, since* $JustArgs_P^{s,w}$ *is contradictory w.r.t.* $\neg a$ *and* a*. Then* $A_{\neg a}^s$ *and* A_a^s *are contradictory*$_P^{s,w}$ *arguments. Arguments* A_b^s*,* A_c^s*,* A_d^s*,* $A_g'^s$ *are based-on-contradiction*$_P^{s,w}$*.*

Note that A_b^s *is built based on the contradictory argument* A_a^s*, i.e.* $a \in Conc(A_b^s)$*.* $A_{\neg a}^s$ *is in both sets of counter-attack:*

$$CA_{A_c^s}(A_a^w, JustArgs_P^{s,w}) \text{ and } CA_{A_d^s}(A_a^w, JustArgs_P^{s,w})$$

Figure 3.3 illustrates that both arguments A_d^s *and* A_c^s *are reinstated by* $A_{\neg a}^s$ *against* A_a^w*. Similar to the previous case,* $A_g'^s$ *is reinstated by* A_a^s*.*

In the following we present properties of the resulting set of acceptable arguments, i.e. $JustArgs_P^{p,o}$. Given that there are four possibilities of interactions between proposed and opposing arguments, the following holds:

Theorem 24 *Let P be an ELPd, $L \in \mathcal{H}(P)$, p (resp. o) be the kind — i.e. strong or weak — of a proposed (resp. opposing) argument of P, and $S_{p,o}$ be a set of arguments such that $\forall A_L^p : A_L^p \in lfp(F_P^{p,o})$. The sets $S_{p,o}$ (of justified arguments) have pre-fixpoint relations:*

i. $S_{w,s} \subseteq S_{w,w}$

ii. $S_{w,s} \subseteq S_{s,s}$

iii. $S_{w,w} \subseteq S_{s,w}$

iv. $S_{s,s} \subseteq S_{s,w}$

Proof. Given that (i) and (iv), and also (ii) and (iii), have similar kinds of proposed arguments and opposing arguments (i.e. p and o, respectively), we prove (i–iv) by proving that:

1. $S_{k,s} \subseteq S_{k,w}$
2. $S_{w,k} \subseteq S_{s,k}$

for $k \in \{s, w\}$.

1. Let $S = lfp(F_P^{k,s})$. We first prove, by contradiction, that S is a pre-fixpoint[6] of $F_P^{k,w}$, i.e. for every L if $A_L^k \in S$ then $A_L^k \in F_P^{k,w}(S)$. Assume, by contradiction, that $A_L^k \in S$ and A_L^k is not acceptable$_{k,w}$ w.r.t. S, i.e. $A_L^k \notin F_P^{k,w}(S)$. So there exists an argument $A_{L'}^w$ attacking A_L^k that it is not attacked by any argument in S. But in this case there also exists an $A_{L'}^s$ attacking A_L^k that, based on Prop. 12, is not attacked by any argument in S. Thus A_L^k is not acceptable$_{k,s}$ w.r.t. S, i.e. $A_L^k \notin S$. Contradiction. So S is a pre-fixpoint of $F_P^{k,w}(S)$. Given that $F_P^{k,w}$ is monotonic (cf. Prop. 16), and that, as is well known, pre-fixpoints of monotonic operators are included in all fixpoints, $S \subseteq lfp(F_P^{k,w})$, i.e. $lfp(F_P^{k,s}) \subseteq lfp(F_P^{k,w})$.

2. Let S be the set of the strong arguments corresponding to the arguments in $lfp(F_P^{w,k})$, A_L^s be an argument in S, and A_L^w be its weak corresponding argument. Given that $Assump(A_L^s) \subseteq Assump(A_L^w)$ and that A_L^w is acceptable$_{w,k}$ w.r.t. $lfp(F_P^{w,k})$, clearly also A_L^s is acceptable$_{s,k}$ w.r.t. S. So, if $A_L^s \in S$, A_L^s also belongs to $F_P^{s,k}(S)$, i.e. S is a pre-fixpoint of $F_P^{s,k}$. So, similarly to point (1), $S \subseteq lfp(F_P^{s,k})$. Accordingly, if there is an argument for L in S (i.e. $L \in S_{w,k}$) then there is also an argument for L in $lfp(F_P^{s,k})$ (i.e. $L \in S_{s,k}$) ■

[6] A set S is a pre-fixpoint of a function φ iff $S \subseteq \varphi(S)$.

The resulting set of acceptable arguments from an argumentation process over a set S of arguments with just weak arguments is always consistent. I.e. neither weak arguments for $\bot$ (called $A^w_\bot$) nor contradictory arguments (i.e. both weak arguments for $\neg L$ and L) are considered acceptable w.r.t. S. Intuitively, if there is an argument $A^w_\bot$ (with a rule $\bot \leftarrow Body$) such that L is in $Body$, we can deduce that $A^w_\bot$ was built based on A^w_L. If we accept A^w_L, we also accept $A^w_\bot$, then *falsity* follows. Therefore, both arguments A^w_L and $A^w_\bot$ are not acceptable. So, the following holds:

Proposition 25 *$JustArgs^{w,w}_P$ is conflict-free.*

Proof. Given that $JustArgs^{w,w}_P = lfp(F^{w,w}_P)$, we prove that $lfp(F^{w,w}_P)$ is conflict-free. Let $F^{w,w}_P(S^0) \subseteq \ldots F^{w,w}_P(S^i) \subseteq \ldots \subseteq F^{w,w}_P(S^\omega) = lfp(F^{w,w}_P)$. We prove, by induction, that all S^i in the sequence are conflict-free.

Base: Clearly, since $S^0 = \emptyset$, S^0 is conflict-free.

Induction: Assume that S^i is conflict-free. By contradiction, further assume that S^{i+1} is not conflict-free, i.e. there exists $\{A^w_{L_1}, A^w_{L_2}\} \subseteq S^{i+1}$ such that $A^w_{L_1}$ attacks $A^w_{L_2}$. Since $A^w_{L_1}$ attacks $A^w_{L_2}$ then, given that $A^w_{L_2}$ is acceptable$_{w,w,}$ w.r.t. S^i (i.e. $A^w_{L_2} \in S^{i+1}$), there must exist $A^w_{L_3} \in S^i$ such that $A^w_{L_3}$ attacks $A^w_{L_1}$. Given that, by induction hypothesis, S^i is conflict-free, no argument in S^i attacks $A^w_{L_3}$. So $A^w_{L_1}$ is not acceptable$_{w,w,}$ w.r.t. S^i, i.e. $A^w_{L_1} \notin S^{i+1}$. Contradiction ■

Proposition 26 *$JustArgs^{w,w}_P$ is non-contradictory.*

Proof. Given that $JustArgs^{w,w}_P = lfp(F^{w,w}_P)$, we prove that $lfp(F^{w,w}_P)$ is non-contradictory. Let $F^{w,w}_P(S^0) \subseteq \ldots F^{w,w}_P(S^i) \subseteq \ldots \subseteq F^{w,w}_P(S^\omega) = lfp(F^{w,w}_P)$. We prove, by induction, that all S^i in the sequence are non-contradictory.

Base: Clearly, since $S^0 = \emptyset$, S^0 is non-contradictory.

Induction: Assuming that S^i is non-contradictory, we must prove that (1) $A^w_\bot \notin S^{i+1}$ and (2) there exists no L such that $\{A^w_L, A^w_{\neg L}\} \subseteq S^{i+1}$.

1. By induction hypothesis, $A^w_\bot \notin S^i$. If there is no argument in S^i attacking $A^w_\bot$ and since $A^w_\bot$ attacks itself, $A^w_\bot$ is not acceptable$_{w,w}$ w.r.t. S^i and so $A^w_\bot \notin S^{i+1}$. If there is an argument $A^w_L \in S^i$ attacking $A^w_\bot$, and since S^i is conflict-free (cf. proof of proposition 25), $A^w_\bot$ is not acceptable$_{w,w}$ w.r.t. S^i, i.e. $A^w_\bot \notin S^{i+1}$.

2. Since, by induction hypothesis, S^i is non-contradictory, then for every literal L either there exists an argument for $A^w_L \in S^i$ and no argument for $A^w_{\neg L} \in S^i$, or there are no arguments in S^i for neither A^w_L nor $A^w_{\neg L}$:

(a) $A_L^w \in S^i$ and $A_{\neg L}^w \notin S^i$. Since A_L^w attacks $A_{\neg L}^w$ and, by proof of proposition 25, A_L^w is not attacked by any argument in S^i, then $A_{\neg L}^w$ is not acceptable$_{w,w}$ w.r.t. S^i, i.e. $A_{\neg L}^w \notin S^{i+1}$; or

(b) $A_L^w \notin S^i$ and $A_{\neg L}^w \notin S^i$. If there is no argument in S^i attacking neither A_L^w nor $A_{\neg L}^w$ then none of them is acceptable$_{w,w}$ w.r.t. S^i because they attack each other. So $A_L^w \notin S^{i+1}$ and $A_{\neg L}^w \notin S^{i+1}$. If there is one such argument $A_{L_1}^w \in S^i$ attacking say A_L^w then A_L^w is not acceptable$_{w,w}$ w.r.t. S^i because it is attacked by $A_{L_1}^w$ which in turn is not attacked by any argument in S^i (cf. proof of proposition 25). Thus $A_L^w \notin S^{i+1}$.

In any of these two cases, (2.a) and (2.b), at least one of A_L^w or $A_{\neg L}^w$ does not belong to S^{i+1} and so, since this was proven for every L, S^{i+1} is non-contradictory ■

To better understand the properties of $JustArgs_P^{p,o}$ consider the following example (a smaller P example than that of Example 10, tailored to illustrate the properties):

Example 20 *Assume the extended logic program*

$$P = \{a \leftarrow not\ \ b;\quad \neg a;\quad b;\quad \neg b;\quad c;\quad \bot \leftarrow c\}$$

The weak arguments of P are as follows:

$$\begin{array}{rcl} A_a^w & : & [a \leftarrow not\ \ b, not\ \ \neg a, not\ \ \bot] \\ A_{\neg a}^w & : & [\neg a \leftarrow not\ \ a, not\ \ \bot] \\ A_b^w & : & [b \leftarrow not\ \ \neg b, not\ \ \bot] \\ A_{\neg b}^w & : & [\neg b \leftarrow not\ \ b, not\ \ \bot] \\ A_c^w & : & [c \leftarrow not\ \ \neg c, not\ \ \bot] \\ A_\bot^w & : & A_c^w + [\bot \leftarrow c, not\ \ \neg\bot, not\ \ \bot] \end{array}$$

The argument A_a^w (resp. A_b^w) attacks $A_{\neg a}^w$ (resp. $A_{\neg b}^w$), and vice-versa. Moreover, $A_\bot^w$ is self-defeating. Finally, A_a^w (resp. A_c^w) is attacked by A_b^w (resp. $A_\bot^w$) and there is no argument to reinstate it. Thus, no argument belongs to $JustArgs_P^{w,w}$. Figure 3.5 illustrates the above description based upon the acceptability argument. See Remark 14 to understand better the notation used in the figure.

The acceptability of arguments of "Privacy of Personal Life" (PPL) is illustrated in Figure 3.6. See Remark 14 on page 48 to understand better the notation used in the figure. Every argument from $Args^w(PPL)$ is in the figure. The figure depicts the attacking relation between those arguments and also the building dependency[7] of such arguments. Some comments about the attacks in $Args^w(PPL)$ are as follows:

[7]By "building dependency" we mean that an argument is built based on other arguments.

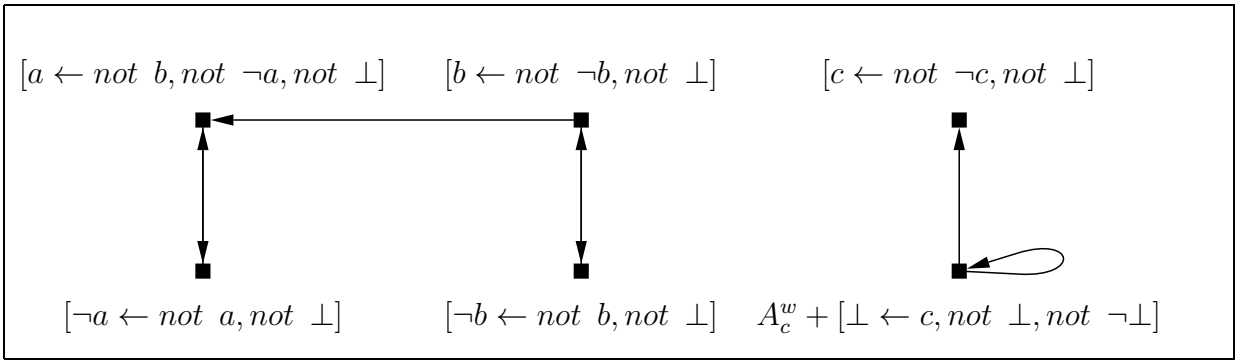


Figure 3.5: Acceptable$_{w,w}$ arguments in $Args^w(P)$

- the arguments $A^w_{hP(i)}$ and $A^w_{\neg hP(i)}$ (resp. $A^w_{hP(a)}$ and $A^w_{\neg hP(a)}$) attack each other. Every weak argument for $hP(p)$ is attacked by $A^w_{\neg hP(p)}$, and vice-versa. Moreover, none of those arguments is reinstated by other argument in $Args^w(PPL)$;

- both conclusions $\neg hP(i)$ and $ev(i, tFDD)$ lead to *falsity* in PPL. Then, the weak argument for $\bot$ attacks both arguments $A^w_{\neg hP(i)}$ and $A^w_{ev(i,tFDD)}$. Moreover, $A^w_\bot$ attacks itself. Thus, none of the these arguments is acceptable$^{w,w}_{PPL}$ (w.r.t. $Args^w(PPL)$);

- the arguments for $pLE(i)$, $pLE(a)$ and $pLE(p)$ are built based on a previous argument that is not acceptable$^{w,w}_{PPL}$ which is $A^w_{\neg hP(i)}$, $A^w_{\neg hP(a)}$, or $A^w_{\neg hP(p)}$, respectively. So, the former arguments are also not acceptable$^{w,w}_{PPL}$ because they are attacked by exactly the same attacking argument as the latter arguments;

- finally, the other arguments in the figure are acceptable$^{w,w}_{PPL}$.

Given that $JustArgs^{w,s}_P \subseteq JustArgs^{w,w}_P$ (cf. Theorem 24), the following holds:

Proposition 27 *$JustArgs^{w,s}_P$ is conflict-free and non-contradictory.*

Proof. Since $lfp(F^{w,s}_P) \subseteq lfp(F^{w,w}_P)$ (cf. Theorem 24) and that $lfp(F^{w,w}_P)$ is conflict-free and non-contradictory (cf. Propositions 25 and 26, respectively), then $JustArgs^{w,s}_P$ is conflict-free and non-contradictory ■

Note that $F^{w,s}_P$ does not consider [ADP95]'s 'Coherence Principle': "If L is explicitly false then L must be assumed false by default". Given that every opposing argument is a strong one, it cannot be attacked by any argument for its explicit negation. In Example 20, the argument A^w_a is attacked by both A^s_b and

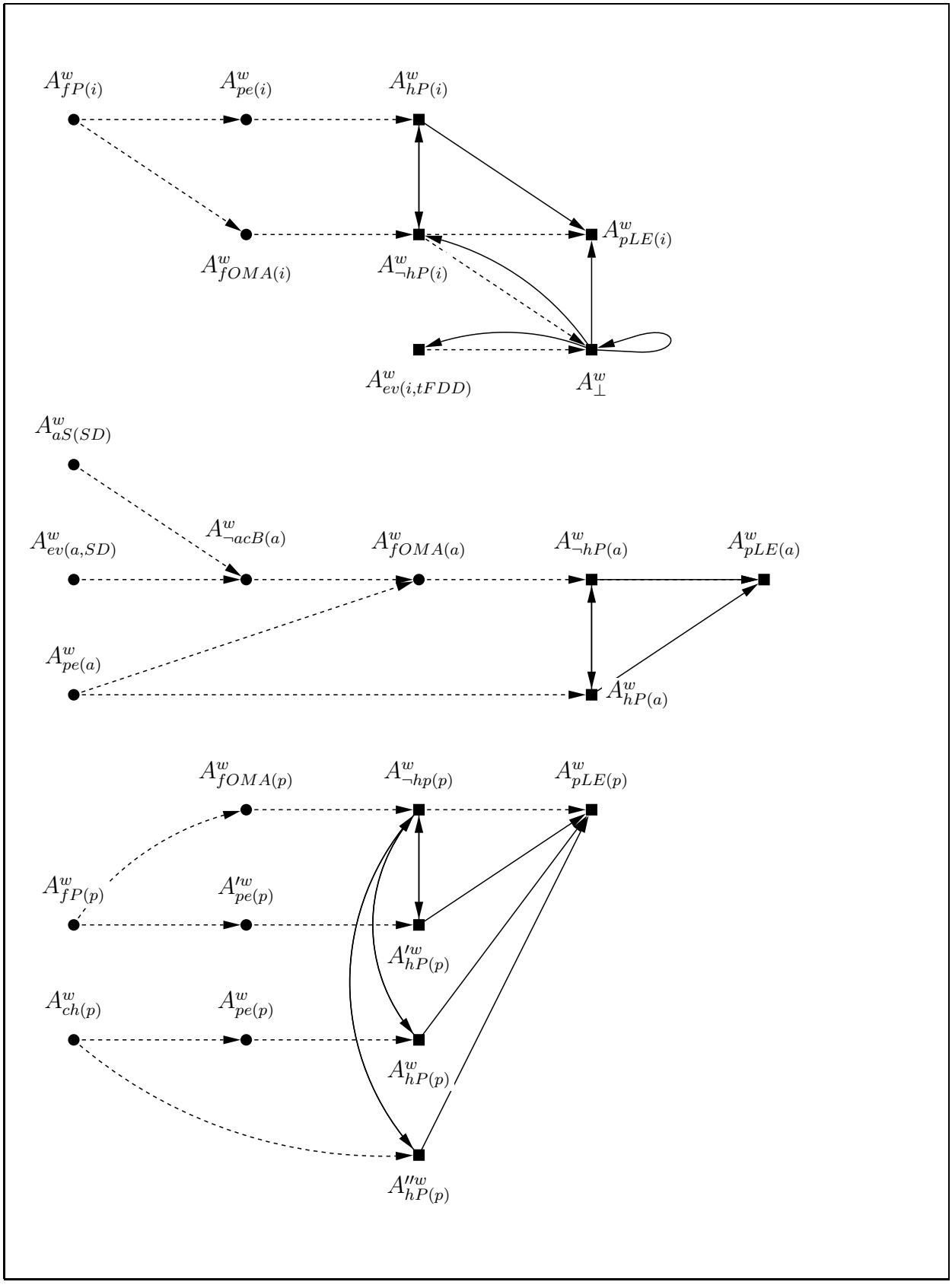


Figure 3.6: Acceptable$_{w,w}$ arguments in $Args^w(PPL)$

$A^s_{\neg a}$ and so it is not acceptable$^{w,s}_P$ w.r.t. $\mathcal{A}rgs(P)$. However, there are arguments in $\mathcal{A}rgs(P)$ that could reinstate such an argument, viz. arguments for both a and $\neg b$. Furthermore, we can say that $F_P^{w,w}$ has more defensible arguments than $F_P^{w,s}$. In other words, $F_P^{w,s}$ has more overruled arguments than $F_P^{w,w}$.

Proposition 28 $gfp(F_P^{w,s}) \subseteq gfp(F_P^{w,w})$.

Proof. Let $S = gfp(F_P^{w,w})$. We first prove, by contradiction, that S is a pos-fixpoint[8] of $F_P^{w,s}$, i.e. for every L if $A^w_L \in F_P^{w,s}(S)$ then $A^w_L \in S$. Assume, by contradiction, that $A^w_L \notin S$ and A^w_L is acceptable$_{w,s}$ w.r.t. S , i.e. $A^w_L \in F_P^{w,s}(S)$. So there exists an argument $A^s_{L'}$ attacking A^w_L that it is attacked by an argument in S. But, in this case, there also exists an $A^w_{L'}$ (corresponding to $A^s_{L'}$) attacking A^w_L that, based on Prop. 12, is also attacked by an argument in S. Thus $A^w_L \in S$. Contradiction. So S is a pos-fixpoint of $F_P^{w,s}$. Given that $F_P^{w,w}$ is monotonic (cf. Prop. 16), and that, as is well known, pos-fixpoints of monotonic operators include all fixpoints, $gfp(F_P^{w,s}) \subseteq S$, i.e. $gfp(F_P^{w,s}) \subseteq gfp(F_P^{w,w})$ ■

At this point we have illustrated that the resulting set of acceptable arguments w.r.t. a set of weak arguments is always consistent. To handle paraconsistency, the set of arguments should have both strong and weak arguments. To evaluate the acceptability of arguments without considering the presence of inconsistency, the proposed arguments should be the strong ones. The weak arguments are used to attack, and so, to guarantee the rebuttal. Note that strong arguments also counter-attack (or reinstate evaluating strong arguments). By defining this attacking relation between strong arguments and weak arguments, we respect [ADP95]'s 'Coherence Principle'. Given that every opposing argument is a weak one, it can be attacked by any argument for its explicit negation. Figure 3.7 illustrates the possible attacks of arguments in $Args(P)$. Consider Example 20: A^s_a is attacked by A^w_b but it is reinstated by $A^s_{\neg b}$. Thus, every argument is acceptable$^{s,w}_P$ w.r.t. $Args(P)$.

Figure 3.8 illustrates the possible attacks of arguments in $Args(PPL)$. The argument $A^s_{hP(a)}$ is not acceptable$^{s,w}_{PPL}$ (w.r.t. $Args(PPL)$) because it is attacked by $A^w_{\neg hP(a)}$ and the only counter-attack is $A^s_{hP(a)}$ (in such a case the argument cannot reinstate itself). Moreover, the argument $A^w_{\neg hP(p)}$ attacks two arguments for $hP(p)$, viz. $A'^s_{hp(p)}$ and $A^s_{hp(p)}$. However, $A''^s_{hp(p)}$ reinstates both arguments by attacking $A^w_{\neg hP(p)}$. Thus, the three strong arguments for $hP(p)$ are acceptable$^{s,w}_{PPL}$ while $A^s_{\neg hP(p)}$ is not. The other strong arguments are acceptable$^{s,w}_{PPL}$ because they are not attacked by any argument in $Args(PPL)$.

Finally, it is possible to have an argumentation process with just (attacking and counter-attacking) strong arguments. Looking at Figures 3.9 and 3.10, it is

[8] A set S is a pos-fixpoint of a function φ iff $\varphi(S) \subseteq S$.

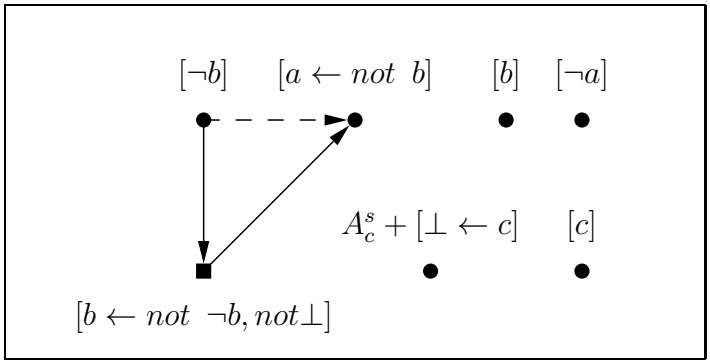


Figure 3.7: Acceptable$_{s,w}$ arguments in $Args(P)$

possible to verify that some arguments are not acceptable because there is no argument to reinstate them.

Given that $JustArgs_P^{w,w}$ and $JustArgs_P^{w,s}$ are non-contradictory sets (by Prop. 27 and Prop. 26, respectively), every argument in both of them is non-contradictory. Nevertheless, $JustArgs_P^{s,w}$ and $JustArgs_P^{s,s}$ may have contradictions.

We may conclude that a justified$_P^{s,w}$ argument is based-on-contradiction$_P^{s,w}$ if the following holds:

Proposition 29 *An argument A^s_L, with L different from $\bot$, is based-on-contradiction$_P^{s,w}$ iff A^s_L is justified$_P^{s,w}$, there does not exist a justified$_P^{s,w}$ argument for $\neg L$ and A^s_L is also overruled$_P^{s,w}$.*

Proof. $\Rightarrow$ Assuming A^s_L is based-on-contradiction$_P^{s,w}$. We have to prove that (1) A^s_L is justified$_P^{s,w}$, (2) $\nexists A^s_{\neg L} \in JustArgs_P^{s,w}$ and (3) A^s_L is also overruled$_P^{s,w}$

1. By definition, any based-on-contradiction$_P^{s,w}$ argument is justified$_P^{s,w}$.

2. Assume, by contradiction, that A^s_L is based-on-contradiction$_P^{s,w}$ and $\exists A^s_{\neg L} \in JustArgs_P^{s,w}$. By definition, $A^s_L \in JustArgs_P^{s,w}$. Given that $\{A^s_L, A^s_{\neg L}\} \subseteq JustArgs_P^{s,w}$, $JustArgs_P^{s,w}$ is contradictory w.r.t. L. Therefore, A^s_L is contradictory$_P^{s,w}$. Contradiction.

3. We have to prove that A^s_L is overruled$_P^{s,w}$. From Theorem 22, A^s_L is overruled$_P^{s,w}$ if the corresponding A^w_L does not belong to $gfp(F_P^{w,s})$. By Lemma 20, $A^w_L \notin gfp(F_P^{w,s})$ when $\exists A^s_{L_1} \in lfp(F_P^{s,w})$ that attacks A^w_L.

   By definition, an argument A^s_L is based-on-contradiction$_P^{s,w}$ iff (a) $\forall A^w_{L'}$ attacking A^s_L there exists a contradictory$_P^{s,w}$ or based-on-contradiction$_P^{s,w}$ $A^s_{L''}$ argument in $CA_{A_L}(A^w_{L'}, JustArgs_P^{s,w})$, or (b) $\exists L' \in Conc(A^s_L)$, different from L, such that $JustArgs_P^{s,w}$ is contradictory w.r.t. L'.

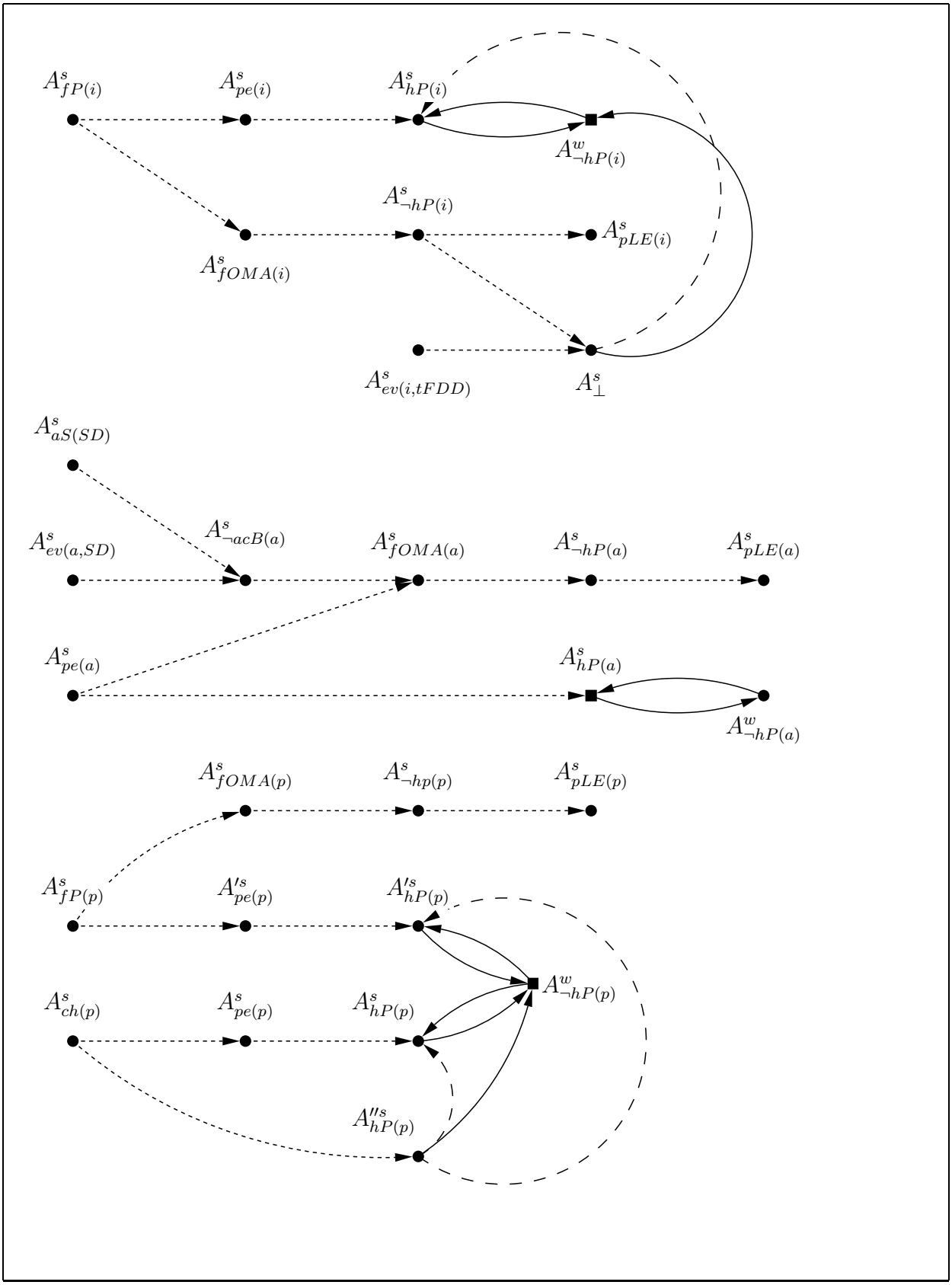


Figure 3.8: Acceptable^{s,w} arguments in $Args(PPL)$

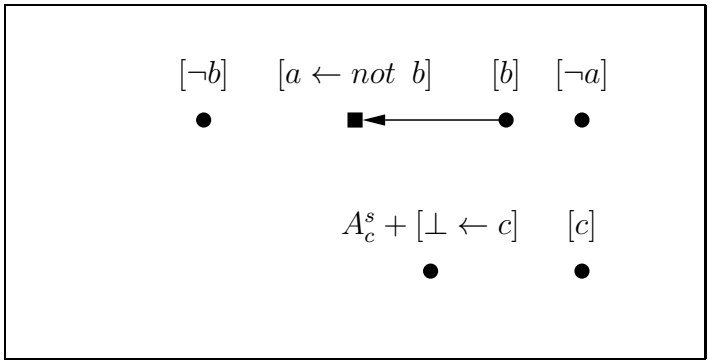


Figure 3.9: Acceptable$_{s,s}$ arguments in $Args^s(P)$

(a) Cf. Prop. 12, if $\exists A^w_{L'}$ attacking A^s_L then $A^w_{L'}$ attacks the corresponding A^w_L. Moreover, $A^s_{L'}$ corresponding to $A^w_{L'}$ also attacks A^w_L. We have to prove that $A^s_{L'} \in lfp(F_P^{s,w})$. By definition, $Assump(A^w_{L'}) - Assump(A^s_{L'}) = S = not\{\bot, \neg L', \neg L_1, \ldots, \neg L_j\}$ such that $L_i \leftarrow Body \in A^s_{L'}$ $(1 \leq i \leq j)$. Assuming that $A^s_{L''}$ is contradictory$_P^{s,w}$, then $\exists \neg L_i \in S$ such that $L'' = \neg L_i$. So, there does not exist any $A^w_{L''}$ attacking $A^s_{L'}$. Consequently, $A^s_{L'} \in lfp(F_P^{s,w})$. Thus, $A^w_L \notin gfp(F_P^{w,s})$ and so A^s_L is overruled$_P^{s,w}$. If $A^s_{L''}$ is based-on-contradiction$_P^{s,w}$ then we use the previous deduction the necessary number of times *q.e.d.*

(b) Assume, by hypothesis,

$$\exists L' \in Conc(A^s_L) \text{ such that } JustArgs_P^{s,w} \text{ is contradictory w.r.t. } L'$$

Accordingly, $\{A^s_{L'}, A^s_{\neg L'}\} \subseteq JustArgs_P^{s,w}$. By definition, there is a rule '$L' \leftarrow Body, not\ \neg L', not\ \bot$' in A^w_L. Therefore, $A^s_{\neg L'} \in lfp(F_P^{s,w})$ such that $A^s_{\neg L'}$ attacks A^w_L, and so A^s_L is overruled$_P^{s,w}$ *q.e.d.*

$\Leftarrow$ Assume that there does not exist a justified$_P^{s,w}$ argument for $\neg L$, and A^s_L is both justified$_P^{s,w}$ and overruled$_P^{s,w}$. We have to prove that A^s_L is based-on-contradiction$_P^{s,w}$. By definition, $A^s_{L''}$ is either contradictory$_P^{s,w}$ or based-on-contradiction$_P^{s,w}$. If $A^s_{L''}$ is contradictory$_P^{s,w}$ then $\exists A^s_{\neg L} \in lfp(F_P^{s,w})$ attacking $A^w_{L''}$. $A^w_L \notin gfp(F_P^{w,s})$ and so A^s_L is overruled$_P^{s,w}$. If $A^s_{L''}$ is based-on-contradiction$_P^{s,w}$ then we use the previous deduction the necessary number of times *q.e.d.* ■

As in [PS97], we may also define the *truth value of a conclusion* as follows:

Definition 49 (Truth Value of a Conclusion) *Let P be an ELPd, and $L \in \mathcal{H}(P)$. A literal L over P is*

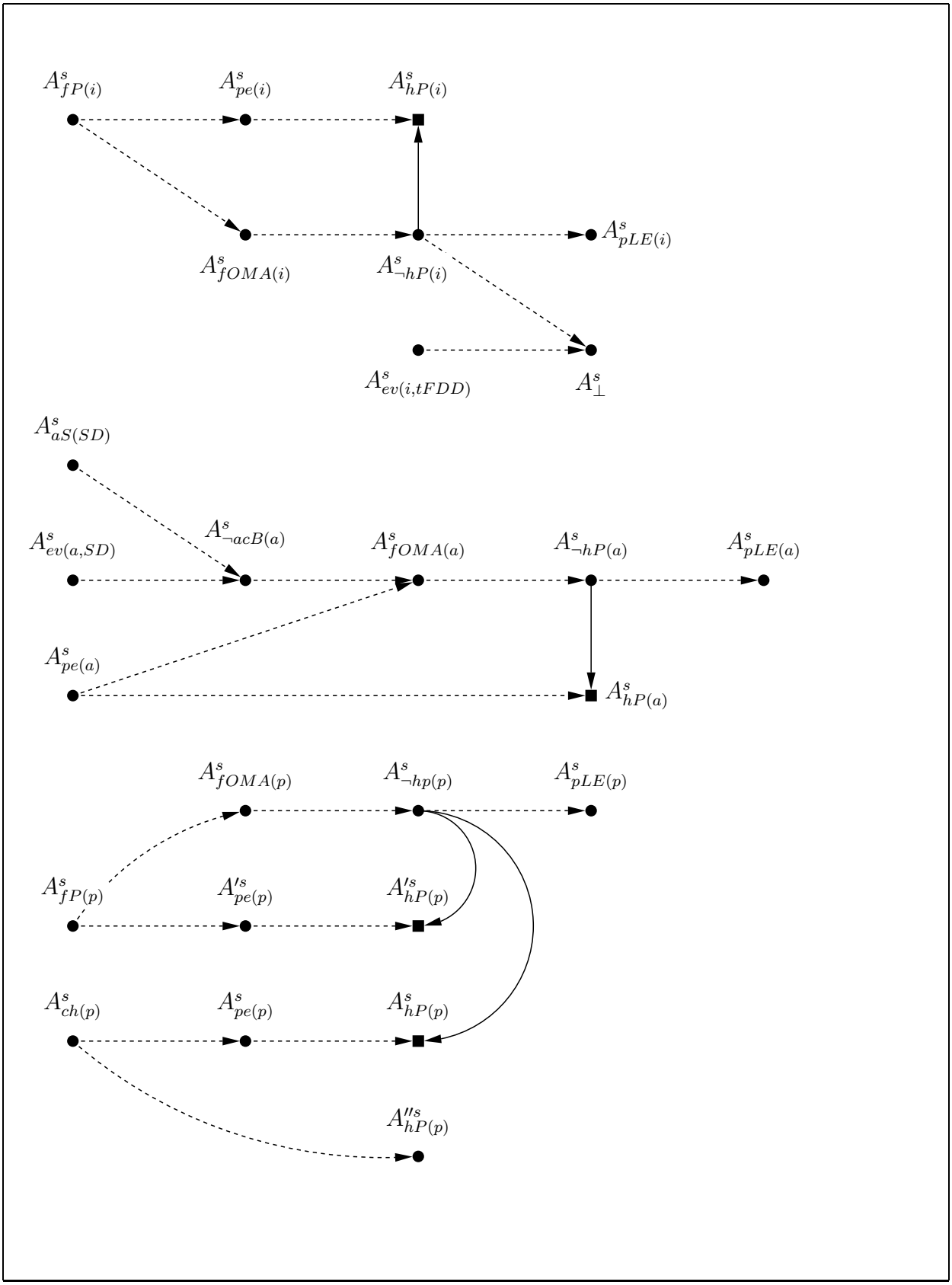


Figure 3.10: Acceptable$_{s,s}$ arguments in $Args^s(PPL)$

- false$_P^{p,o}$ *iff every p-argument for L is overruled*$_P^{p,o}$*;*
- true$_P^{p,o}$ *iff there exists a justified*$_P^{p,o}$ *argument for L;*
- undefined$_P^{p,o}$ *iff L is neither true*$_P^{p,o}$ *nor false*$_P^{p,o}$ *(i.e. there is no justified*$_P^{p,o}$ *argument for L and at least one p-argument for L is not overruled*$_P^{p,o}$*).*

First let us define the status of a literal w.r.t. contradiction, something that follow in the obvious way that of arguments (cf. Definition 48).

Definition 50 (Literals' relation to contradiction) *Let P be an ELPd, and let* $L \in \mathcal{H}(P)$ *be a true*$_P^{p,o}$*. We say that:*

- *L is contradictory*$_P^{p,o}$ *iff there is an argument* A_L^p *which is contradictory*$_P^{p,o}$*;*
- *L is based-on-contradiction*$_P^{p,o}$ *iff it is not contradictory*$_P^{p,o}$ *and there is an argument* A_L^p *which is based-on-contradiction*$_P^{p,o}$*;*
- *L is non-contradictory*$_P^{p,o}$*, otherwise.*

Given that every $JustArgs_P^{w,w}$ and $JustArgs_P^{w,s}$ arguments are non-contradictory (cf. Prop 26 and 27), if L is true$_P^{w,k}$ ($k \in \{s, w\}$) then L is non-contradictory. The same does not hold for $JustArgs_P^{s,w}$ and $JustArgs_P^{s,s}$. However,in the case of $JustArgs_P^{s,w}$, it is easy to check when a literal is contradictory, based on contradiction, or non-contradictory.

Proposition 30 *A literal L over P is*

- *contradictory*$_P^{s,w}$ *if L is the symbol* $\bot$ *or* $\neg L$ *is true*$_P^{s,w}$*;*
- *based-on-contradiction*$_P^{s,w}$ *if it is also false*^{s,w}*;*
- *non-contradictory*$_P^{s,w}$*, otherwise.*

Proof.

contradictory$_P^{s,w}$

The case for $L = \bot$ is trivial, by definition. If L is true$_P^{s,w}$ and $\neg L$ is also true$_P^{s,w}$, then by definition of true$_P^{p,o}$ and of contradictory$_P^{p,o}$ arguments, there exists an argument A_L^P which is contradictory$_P^{p,o}$ *q.e.d.*.

based-on-contradiction$_P^{s,w}$

If L is true$_P^{s,w}$ *then* A_L^s *is justified*$_P^{s,w}$*, by definition. If, furthermore, L is* $false_P^{s,w}$*, then, by definition, every* A_P^s *is overruled*$_P^{s,w}$*. Since, by assumption in*

this proposition L is not contradictory$_P^{s,w}$ then $\neg\exists A^s_{\neg L} \in JustArgs_P^{s,w}$. So, we are in the conditions of Proposition 29, and can conclude that there is an A^s_L which is based-on-contradiction$_P^{s,w}$, and so L is based-on-contradiction$_P^{s,w}$, by definition. q.e.d.

non-contradictory$_P^{s,w}$

Otherwise, L can only be non-contradictory$_P^{s,w}$ ■

Example 21 *Following both Examples 17 and 19, which present the status of arguments in $\mathcal{A}rgs(P)$ and the relation to contradiction of the justified arguments, respectively, the objective literals a, $\neg a$, b, c, d, f and g are true$_P^{s,w}$. Moreover, a and $\neg a$ are contradictory$_P^{s,w}$, b, c and d are based-on-contradiction$_P^{s,w}$, and f and g are non-contradictory$_P^{s,w}$. Since every argument for a, $\neg a$, h, e, c, d, and b is overruled$_P^{s,w}$, these objective literal are false$_P^{s,w}$. The undefined$_P^{s,w}$ objective literals are i and j.*

If we look at Figures 3.6, 3.8 and 3.10 we get both the status of the arguments and the truth value of the conclusions of PPL. The results obtained from this proposal are presented in Tables 3.2, 3.3 and 3.4 (see Remarks 31 and 32).

Remark 31 (Notation for Tables 3.2 and 3.3) *Let p (resp. o) be the kind (strong or weak) of proposed (resp. opposing) arguments. C, BC, NC, OV and D denote, respectively, the* status of an argument *A: 'contradictory', 'based on contradiction', 'non-contradictory', 'overruled' and 'defensible'. By "A^p_L* is *N^p_o" we mean "the status of A^p_L is N in a set of p- proposed arguments and o-opposing arguments".*

Remark 32 (Notation for Table 3.4) *Let p (resp. o) be the kind (strong or weak) of proposed (resp. opposing) argument. C, BC, NC, F, and U denote the* truth value of a conclusion *L: 'true and contradictory', 'true and based on contradiction', 'true and non contradictory', 'false', and 'undefined', respectively. By "L* is *N^p_o" we mean "the truth value of L is N in a set of p- proposed arguments and o- opposing arguments". Since columns U^s_w and U^s_s would have no elements, we omit them from the table.*

In the remainder of this section, we relate our semantics to $WFSX_p$ [ADP95], the grounded (skeptical) extension [Dun95], $WFSX$ [PA92], and WFS [Prz90].

A *paraconsistent way of reasoning* is obtained through $F_P^{s,w}$ and we proceed by deducing the same truth value of objective literals as the $WFSX_p$ semantics:

	C_w^s	BC_w^s	NC_w^s	OV_w^s	D_w^s	C_s^s	BC_s^s	NC_s^s	OV_s^s	D_s^s
$A^s_{fP(i)}$			√					√		
$A^s_{pe(i)}$			√					√		
$A^s_{fOMA(i)}$			√					√		
$A^s_{hP(i)}$	√								√	
$A^s_{\neg hP(i)}$	√					√				
$A^s_{\bot}$	√					√				
$A^s_{ev(i,tFDD)}$	√					√				
$A^s_{pLE(i)}$		√		√			√			
$A^s_{aS(SD)}$			√					√		
$A^s_{ev(a,SD)}$			√					√		
$A^s_{pe(a)}$			√					√		
$A^s_{\neg acB(a)}$			√					√		
$A^s_{fOMA(a)}$			√					√		
$A^s_{pLE(a)}$			√					√		
$A^s_{\neg hP(a)}$			√					√		
$A^s_{hP(a)}$				√					√	
$A^s_{hP(p)}$	√								√	
$A^s_{hP'(p)}$	√								√	
$A^s_{hP''(p)}$	√					√				
$A^s_{fP(p)}$			√					√		
$A^s_{ch(p)}$			√					√		
$A^s_{fOMA(p)}$			√					√		
$A^s_{pe(p)}$			√					√		
$A^{\prime s}_{pe(p)}$			√					√		
$A^s_{\neg hP(p)}$	√					√				
$A^s_{pLE(p)}$		√		√			√			

Table 3.2: The status of arguments w.r.t $Args(PPL)$ and $Args^s(PPL)$.

	C_w^w	BC_w^w	NC_w^w	OV_w^w	D_w^w
$A^w_{fP(i)}$			$\surd$		
$A^w_{pe(i)}$			$\surd$		
$A^w_{fOMA(i)}$			$\surd$		
$A^w_{hP(i)}$					$\surd$
$A^w_{\neg hP(i)}$					$\surd$
$A^w_{\bot}$					$\surd$
$A^w_{ev(i,tFDD)}$					$\surd$
$A^w_{pLE(i)}$					$\surd$
$A^w_{aS(SD)}$			$\surd$		
$A^w_{ev(a,SD)}$			$\surd$		
$A^w_{pe(a)}$			$\surd$		
$A^w_{\neg acB(a)}$			$\surd$		
$A^w_{fOMA(a)}$			$\surd$		
$A^w_{pLE(a)}$					$\surd$
$A^w_{\neg hP(a)}$					$\surd$
$A^w_{hP(a)}$					$\surd$
$A^w_{hP(p)}$					$\surd$
$A^w_{hP'(p)}$					$\surd$
$A^w_{hP''(p)}$					$\surd$
$A^w_{fP(p)}$			$\surd$		
$A^w_{ch(p)}$			$\surd$		
$A^w_{fOMA(p)}$			$\surd$		
$A^w_{pe(p)}$			$\surd$		
$A'^w_{pe(p)}$			$\surd$		
$A^w_{\neg hP(p)}$					$\surd$
$A^w_{pLE(p)}$					$\surd$

Table 3.3: The status of arguments w.r.t $Args^w(PPL)$

	C_w^s	BC_w^s	NC_w^s	F_w^s	C_s^s	BC_s^s	NC_s^s	F_s^s	U_w^w	NC_w^w
$fP(i)$			√				√		√	
$pe(i)$			√				√		√	
$fOMA(i)$			√				√		√	
$hP(i)$	√			√				√		√
$\neg hP(i)$	√			√	√					√
$\bot$	√			√	√					√
$ev(i, tFDD)$		√		√	√					√
$pLE(i)$		√		√		√				√
$aS(SD)$			√				√		√	
$ev(a, SD)$			√				√		√	
$pe(a)$			√				√		√	
$\neg acB(a)$			√				√		√	
$fOMA(a)$			√				√		√	
$\neg hP(a)$			√				√			√
$hP(a)$				√				√		√
$pLE(a)$			√				√			√
$hP(p)$	√			√	√					√
$fP(p)$			√				√		√	
$ch(p)$			√				√		√	
$fOMA(p)$			√				√		√	
$pe(p)$			√				√		√	
$\neg hP(p)$	√			√	√					√
$pLE(p)$		√		√		√				√

Table 3.4: The truth value of PPL's conclusions

Theorem 33 ($WFSX_p$ semantics vs $F_P^{s,w}$) *Let P be an ELP such that $\bot \notin \mathcal{H}(P)$, and let L be an objective literal in $\mathcal{H}(P)$.*

- *$L \in WFSX_p(P)$ iff L is $true_P^{s,w}$;*
- *$not\ \ L \in WFSX_p(P)$ iff L is $false_P^{s,w}$;*
- *$\{L, not\ \ L\} \cap WFSX_p(P) = \emptyset$ iff L is $undefined_P^{s,w}$.*

Proof. We have to prove that (1) $L \in WFSX_p(P)$ iff L is $\text{true}_P^{s,w}$, and (2) $not\ \ L \in WFSX_p(P)$ iff L is $\text{false}_P^{s,w}$.

1. We prove, by induction on the iteration of both $\Gamma\Gamma_s$ and $F_P^{s,w}$, that

$$\forall n : L \in \Gamma\Gamma_s{}^{\uparrow n}(\emptyset) \Leftrightarrow \exists A_L^s \in F_P^{s,w\ \uparrow n}(\emptyset)$$

   Base: Clearly, since $n = 0$, there does not exist neither an $L \in \Gamma\Gamma_s^{\uparrow 0}$ nor $A_L^s \in F_P^{s,w\ \uparrow 0}$.

   Induction: Assume there exists a rule $L \leftarrow not\ \ L_1, \ldots, not\ \ L_j$ in P and $\forall_{m \leq n} : L \in \Gamma\Gamma_s{}^{\uparrow n}(\emptyset)$ iff $A_L^s = [L \leftarrow not\ \ L_1, \ldots, not\ \ L_j] \in F_P^{s,w\ \uparrow n}(\emptyset)$.

   $\Rightarrow$ Assume further that $L \in \Gamma\Gamma_s{}^{\uparrow m+1}(\emptyset)$. We have to prove that $A_L^s \in F_P^{s,w\ \uparrow m+1}(\emptyset)$. If $L \in \Gamma\Gamma_s{}^{\uparrow m+1}(\emptyset)$ then there does not exist any $L_i \in \Gamma\Gamma_s{}^{\uparrow m}(\emptyset)$ $(1 \leq i \leq j)$, i.e. there does not exist a rule $L_i \leftarrow Body \in P$ such that $Body \subseteq \Gamma\Gamma_s{}^{\uparrow m}(\emptyset)$. If $L \in \Gamma\Gamma_s^{\uparrow m}(\emptyset)$ then $A_L^s \in F_P^{s,w\ \uparrow m}(\emptyset)$. So, there does not exist any $A_{L_i}^s \in F_P^{s,w\ \uparrow m}(\emptyset)$. Therefore, $A_L^s \in F_P^{s,w\ \uparrow m+1}(\emptyset)$

   $\Leftarrow$ Assume further that $A_L^s \in F_P^{s,w\ \uparrow m+1}(\emptyset)$. We have to prove that $L \in \Gamma\Gamma_s{}^{\uparrow m+1}(\emptyset)$. If $A_L^s \in F_P^{s,w\ \uparrow m+1}(\emptyset)$ then (1) there does not exist any $A_{L_i}^w \in \mathcal{A}rgs(P)$, i.e. there is no rule $L_i \leftarrow Body$ in P, or (2) there exists an $A_{L_i}^w \in \mathcal{A}rgs(P)$ attacking A_L^s that it is attacked by an argument $A_{L'}^s \in F_P^{s,w\ \uparrow m}(\emptyset)$, i.e. there is an $not\ \ L' \in Assump(A_{L_i}^w)$. If $A_L^s \in F_P^{s,w\ \uparrow m}(\emptyset)$ then $L \in \Gamma\Gamma_s{}^{\uparrow m}(\emptyset)$. So, there does not exist any $L_i \in \Gamma\Gamma_s{}^{\uparrow m}(\emptyset)$. Therefore, $L \in \Gamma\Gamma_s{}^{\uparrow m+1}(\emptyset)$.

2. By definition $not\ \ L \in WFSX_p(P)$ iff $L \notin \Gamma_s(lfp(\Gamma\Gamma_s))$. Since, as proven in point(1), the $lfp(\Gamma\Gamma_s)$ exactly corresponds to $F_P^{s,w}$, this amounts to prove that L is $\text{false}_P^{s,w}$ iff there is an argument in $F_P^{s,w}$ attacking A_L^s. But this means that A_L^s is $\text{overruled}_P^{s,w}$, and so, by definition L is $\text{false}_P^{s,w}$ ■

A *consistent way of reasoning* is obtained through $F_P^{w,w}$. When considering a consistent logic program, $F_P^{w,w}$ coincides with both $WFSX$ semantics [PA92] and [Dun95]'s grounded (skeptical) extension.

Corollary 34 *Let P be a consistent program resulting from a union of extended logic programs. Since $WFSX$ coincides with $WFSX_p$ for consistent programs, then $WFSX$ also coincides with the results of $F_P^{w,w}$* ■

To show that $F_P^{w,w}$ and [Dun95]'s grounded (skeptical) extension coincide, we first relate [Dun95]'s definitions of both *RAA-attack* and *g-attack* to our definition of *attack* as follows:

Lemma 35 *Let P be an ELP such that $\bot \notin \mathcal{H}(P)$, $\{(A_L, L), (A_{L'}, L'), (A_{\neg L}, \neg L)\} \subseteq Args(P)$ be such that not $L \in A_{L'}$, and $\{A_L^w, A_{L'}^w, A_{\neg L}^w\} \subseteq \mathcal{A}rgs(P)$ be such that not $L \in Assump(A_{L'})$. If (A_L, L)* g-attacks *$(A_{L'}, L')$ then A_L^w* attacks *$A_{L'}^w$. If $(A_{\neg L}, \neg L)$* RAA-attacks *(A_L, L) then $A_{\neg L}^w$* attacks *A_L^w.*

Proof. This results follows directly from the construction of weak arguments ■

Theorem 36 (Grounded extension vs $F_P^{w,w}$) *Let P be an ELP such that $\bot \notin \mathcal{H}(P)$, and let L be an objective literal in $\mathcal{H}(P)$.*

- *an argument $(A_L, L) \in lfp(F)$ iff $\exists A_L^w \in lfp(F_P^{w,w})$;*
- *an argument $(\{not\ \ L\}, L) \in lfp(F)$ iff $\neg\exists A_L^w \in gfp(F_P^{w,w})$.*

Proof. We have to prove that (1) an argument $(A_L, L) \in lfp(F)$ iff $\exists A_L^w \in lfp(F_P^{w,w})$, and (2) an argument $(\{not\ L\}, L) \in lfp(F)$ iff $\neg\exists A_L^w \notin gfp(F_P^{w,w})$. We prove, by induction on the iteration of both F and $F_P^{w,w}$, that

1. $\forall n : L \in F\ ^{\uparrow n} \Leftrightarrow \exists A_L^w \in F_P^{w,w\ \uparrow n}$.

   Base: Clearly, since $n = 0$, there does not exist neither an $L \in F^{\uparrow 0}(\emptyset)$ nor an $A_L^w \in F_P^{w,w\ \uparrow 0}(\emptyset)$.

   Induction: Assume that there exists a rule $L \leftarrow not\ \ L_1, \ldots, not\ \ L_j \in P$ and $\forall n : (A, L) = (\{not\ \ L_1, \ldots, not\ \ L_j\}, L) \in F^{\uparrow n}(\emptyset)$ iff $A_L^w = [L \leftarrow not\ \neg L, not\ \bot, not\ \ L_1, \ldots, not\ \ L_j] \in F_P^{w,w\ \uparrow n}(\emptyset)$.

   $\Rightarrow$ Assume further that $(A, L) \in F^{\uparrow n+1}(\emptyset)$. We have to prove that $A_L^w \in F_P^{w,w\ \uparrow n+1}(\emptyset)$. If $(A, L) \in F^{\uparrow n}(\emptyset)$ then there does not exist any $(A_i, L_i) \in F^{\uparrow n}(\emptyset)$ $(1 \leq i \leq j)$, i.e. there does not exist any (A_i, L_i) attacking (A, L) because (1) there is no rule $L_i \leftarrow Body_i \in P$ or (2) there is an $(A_i, L_i) \in AR(P)$ that it is attacked by an argument in $F^{\uparrow n}(\emptyset)$. If $(A, L) \in F^{\uparrow n}(\emptyset)$ then $A_L^w \in F_P^{w,w\ \uparrow n}(\emptyset)$. So, there does not exist any $A_{L_i}^w \in F_P^{w,w\ \uparrow n}(\emptyset)$. Therefore, $A_L^w \in F_P^{w,w\ \uparrow n+1}(\emptyset)$

$\Leftarrow$ Assume further that $A_L^w \in F_P^{w,w\ \uparrow n+1}(\emptyset)$. We have to prove that $(A, L) \in F^{\uparrow n+1}(\emptyset)$. If $A_L^w \in F_P^{w,w\ \uparrow n+1}(\emptyset)$ then there does not exist any $A_{L_i}^w \in F_P^{w,w\ \uparrow n}(\emptyset)$ because (1) there is no rule $L_i \leftarrow Body \in \mathcal{A}rgs(P)$ or (2) there exist an $A_{L_i}^w \in F_P^{w,w\ \uparrow n}(\emptyset)$. If $A_L^w \in F_P^{w,w\ \uparrow n}(\emptyset)$ then $(A, L) \in F^{\uparrow n}(\emptyset)$. So there does exist any $(A_i, L_i) \in F^{\uparrow n}(\emptyset)$. Therefore, $(A, L) \in F^{\uparrow n+1}(\emptyset)$.

2. Given the results of Lemmas 20 and 21, relating greatest and least fixpoints of $F_P^{p,o}$, and given that in the case $p = o = w$, the result follows similarly to the proof in point (1) ■

Finally, $F_P^{s,s}$ coincides with WFS with "classical negation" [Prz90].

Theorem 37 (WFS semantics vs $F_P^{s,s}$) *Let P be an ELP such that $\bot \notin \mathcal{H}(P)$, and let L be an objective literal in $\mathcal{H}(P)$.*

- $L \in WFS(P)$ *iff* L *is* $true_P^{s,s}$
- *not* $L \in WFS(P)$ *iff* L *is* $false_P^{s,s}$
- $\{L, not\ L\} \cap WFS(P) = \emptyset$ *iff* L *is* $undefined_P^{s,s}$.

Proof. This proof is similar to that of Theorem 33. Note that the construction of WFS is obtained by the iteration of the operator $\Gamma\Gamma$, where no semi-normality (i.e. weak arguments) comes in place ■

3.3 Proof for an Argument

> "Something is proved (a literal, an argument, or something else) if there exists at least one proof that succeeds. Something is not proved if every proof fails. In other words, a proof of something fails if every proof fails." [PS97].

Event though the declarative semantics just exposed relies on an iterative procedure, its usage for computing arguments may not always be appropriate. This is especially the case when we are only interested in the proof for a (query) argument, rather than all acceptable arguments, as is obtained by the iterative process. Such a query oriented proof procedure can be viewed as conducting a "dispute between a proponent player and an opponent player" in which both proponent and opponent exchange arguments. In its simplest form, the dispute can be viewed as a sequence of alternating arguments:

$$PR_1, OP_2, PR_3, \ldots, PR_i, OP_{i+1}, PR_{i+2}, \ldots$$

The proponent puts forward an initial argument PR_1. For every argument PR_i put forward by the proponent, the opponent responds with an attacking argument OP_{i+1} against PR_i. For every attacking argument O_i, put forward by the opponent, the proponent attempts to counter-attack with a proposing argument P_{i+1}. To win the dispute, the proponent needs to have a proposed argument against every opposing argument of the opponent. Therefore, a winning dispute can be represented as a dialogue tree, which represents the top-down, step-by-step construction of a proof tree. We follow [PS97]'s proposal, which defines a proof for an argument A_L as a dialogue tree for A_L. However, our definition of dialogue tree is in accordance with the acceptability of the arguments of an ELPd P (see Def. 42):

> A proposed argument $A_L \in \mathcal{A}rgs^p(P)$ is acceptable if all of its opposing arguments in $\mathcal{A}rgs^o(P)$ are attacked by acceptable arguments from $\mathcal{A}rgs^p(P)$.

To define a dialogue tree for an argument A_L we first need a definition of *dialogue for an argument*. A dialogue for A_L is a sequence of PR and OP moves of proposed and opposing arguments, such that the first PR move is the argument A_L. Each OP (resp. PR) move of a dialogue consists of an argument from $\mathcal{A}rgs^o(P)$ (resp. $\mathcal{A}rgs^p(P)$) attacking the previous proposed (resp. opposing) argument. Intuitively, we can say that every PR move wants the conclusion of A_L to be acceptable, and each OP move only wants to prevent the conclusion of A_L from being acceptable. In the case of PR moves, we can further say that if we impose the restriction that proposing arguments cannot be used more than once in a dialogue, then the dialogue will have a finite sequence of PR and OP moves. While none of the proposed arguments can be used more than once in the same dialogue, any of the opposing arguments can still be repeated as often as required to attack proposed arguments.

Definition 51 (*dialogue*$^{p,o}_{A_L}$) *Let P be an ELPd, p (resp. o) be the kind (strong or weak) of a proposed (resp. an opposing) argument of P, and $\mathcal{A}rgs^p(P)$ and $\mathcal{A}rgs^o(P)$ be the set of p-arguments and o-arguments of P, respectively. A* dialogue *p o (in P) for an argument $A_L \in \mathcal{A}rgs^p(P)$, denoted dialogue$^{p,o}_{A_L}$, is a finite non-empty sequence of m moves $move_i = A_{L_i}$ $(1 \leq i \leq m)$ such that*

1. *$move_1 = A_L$;*
2. *for every $1 < i \leq m$, A_{L_i} attacks $A_{L_{i-1}}$ and*
   - *if i is odd then $A_{L_i} \in \mathcal{A}rgs^p(P)$ and there is no odd $j < i$ such that $A_{L_j} = A_{L_i}$, or*
   - *if i is even then $A_{L_i} \in \mathcal{A}rgs^o(P)$.*

We say that $move_i$ is odd if i is odd; otherwise, $move_i$ is even.

A dialogue for A_L *succeeds* if its last move is a PR move. In this proposal, we want to guarantee that a dialogue tree for an argument A_L is finitary, i.e. that the iterative process above is guaranteed to terminate after an enumerable number of steps. Note that we only consider grounded finite ELPd in the declarative semantics (presented in the previous section). By considering this, every dialogue in a dialogue tree finishes because there will always be a last move PR (resp. OP), so no opposing (resp. proposed) argument against it exists.

Definition 52 (Completed, Failed and Successful Dialogue)
Let P be an ELPd. A dialogue p o (in P) for an argument $A_L \in \mathcal{A}rgs^p(P)$ is completed *iff its last move is m, and*

- *if m is odd then there is no argument in $\mathcal{A}rgs^o(P)$ attacking A_{L_m}, or*
- *if m is even then there is no argument in $\mathcal{A}rgs^p(P) - S_p$ attacking A_{L_m}, where S_p is the set of all A_{L_j} in the sequence such that j is odd.*

A completed dialogue is failed *iff its last move is odd; otherwise, it* succeeds.

Remark 38 *From now on, and unless otherwise stated, we refer to 'completed dialogue' simply as 'dialogue'.*

Note that a dialogue$^{p,o}_{A_L}$ in P and the $lfp(F^{p,o}_P)$ "grow" in different ways. In the former, an argument A in the last move, $move_m$, is not attacked by any argument in $\mathcal{A}rgs(P)$. Since A attacks the previous move, $move_{m-1}$, we can say that the argument B in $move_{m-2}$ was reinstated by A. Thus, each $move_i$ $(1 \leq i < m-1)$ is reinstated by $move_{i+2}$. The latter evaluates argument A as acceptable in the first iteration of the characteristic function $F^{p,o}_P$. In the second iteration, A reinstates an argument B, so that B is acceptable and may reinstate other arguments in all following iterations. We can further say that dialogue$^{p,o}_{A_L}$ decreases (in a top-down way) and $lfp(F^{p,o}_P)$ increases (in a bottom-up way) the set of evaluated arguments.

Proposition 39 *Let $move_m = A_L$ be the last move of a succeeded dialogue$^{p,o}_{A_{L'}}$ in P. Then, $A_L \in F^{p,o}_P(\emptyset)$.*

Proof. We have to prove that A_L of the last move of a succeeded dialogue$^{p,o}_{A_{L'}}$ is an acceptable$_{p,o}$ argument with respect to $\mathcal{A}rgs(P)$. Obvious, if A_L is the argument of the last move of a succeeded dialogue$^{p,o}_{A_{L'}}$, then A_L is a p-argument and there is no o-argument attacking it. If A_L is a p-argument and there is no o-argument attacking it, then $A_L \in F^{p,o}_P(\emptyset)$ ■

A dialogue tree DT for A_L considers all possible ways in which A_L can be attacked. The tree has root A_L and each branch of DT is a dialogue for A_L. Furthermore, every single dialogue for A_L has a corresponding branch in the tree because we must consider all the arguments in $\mathcal{A}rgs(P)$ to deduce the status of A_L. The dialogue tree DT for an argument A_L succeeds if every dialogue of DT succeeds.

Definition 53 ($DT^{p,o}_{A_L}$) *Let P be an ELPd, p (resp. o) be the kind (strong or weak) of the proposed (resp. opposing) argument of $\mathcal{A}rgs(P)$, and $\mathcal{A}rgs^p(P)$ (resp. $\mathcal{A}rgs^o(P)$) be the set of p-arguments (o-arguments) of P. A* dialogue tree *p o (in P) for $A_L \in \mathcal{A}rgs^p(P)$, denoted $DT^{p,o}_{A_L}$, is a finite tree of m moves $move_i = A_{L_i}$ $(1 \leq i \leq m)$ such that*

1. *each branch is a dialogue$^{p,o}_{A_L}$, and*
2. *for all i, if $move_i$ is*
   - *even then its only child is a p-argument attacking $A_{L_i} \in \mathcal{A}rgs^o(P)$;*
   - *odd then its children are all o-arguments attacking $A_{L_i} \in \mathcal{A}rgs^p(P)$*

A branch of the tree succeeds if it corresponds to a succeeded dialogue. A $DT^{p,o}_{A_L}$ succeeds *iff all branches (i.e. all dialogue$^{p,o}_{A_L}$) of the tree succeed.*

According to the second condition of Definition 53, we may obtain more than one dialogue tree for an argument. This occurs because only one proponent's move is considered for each opponent's move of a dialogue tree. The following Example illustrates this.

Example 22 *Let P be an ELPd as follows*

$$\{p \leftarrow not\ \ a;\ a \leftarrow not\ \ b, not\ \ c;\ a \leftarrow not\ \ d;\ b; d \leftarrow not\ \ e;\ c \leftarrow not\ \ g;\ g\}$$

Figure 3.11 presents the two possible $DT^{s,s}_{A_p}$ in P. The first dialogue tree succeeds, and the second one does not succeed because there is a failed dialogue (i.e. there is a last move with an o-argument: $move_4 = A^o_g = [g]$).

At this point we can relate the results from a $DT^{p,o}_{A_L}$ to the status of the argument A_L (see Def. 44), as follows:

Proposition 40 *An argument A^p_L in P is*

- *justified$^{p,o}_P$ iff there exists a successful $DT^{p,o}_{A_L}$;*
- *overruled$^{p,o}_P$ iff for all $DT^{p,o}_{A_L}$ there exists a $move_2 = A^o_{L'}$ such that $DT^{o,p}_{A_{L'}}$ succeeded;*

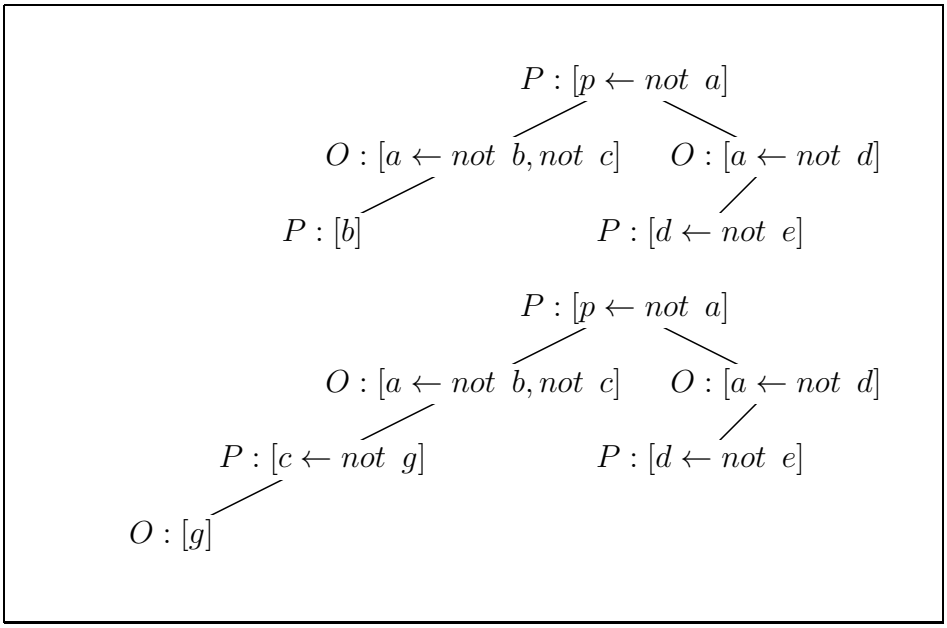


Figure 3.11: $DT^{s,s}_{A_p}$ in $\{p \leftarrow not\ a;\ a \leftarrow not\ b, not\ c;\ a \leftarrow not\ d;\ b;\ d \leftarrow not\ e;\ c \leftarrow not\ g;\ g\}$

- *defensible*$^{p,o}_P$ *iff it is neither justified*$^{p,o}_P$ *nor overruled*$^{p,o}_P$.

Proof. Below we need a notion of size of dialogue tree. The size of a dialogue tree $DT^{p,o}_{A_L}$, $size(DT^{p,o}_{A_L})$, is the maximum of the sizes of its dialogues. The size of a dialogue is determined by the number of odd moves.

In the remainder of the proof, we denote by DT_{A_L} (resp. $size(DT_{A_L})$, A_L, $A_\perp$, F_P) the $DT^{p,o}_{A_L}$ (resp. $size(DT^{p,o}_{A_L})$, A^p_L, $A^p_\perp$, $F^{p,o}_P$).

Justified arguments for L

The proof for $justified^{p,o}_P$ arguments for L follows by induction on the size of DT_{A_L} and the number of iterations of F_P starting from $\emptyset$, needed to obtain A_L.

Formally, we prove by induction on n that:

(1) $A_L \in F_P^{\uparrow n}$ iff there exists a successful DT_{A_L} such that

$$size(DT_{A_L}) \leq n$$

Base: $A_L \in F_P^{\uparrow 1}$ iff there is no $A^o_{L'}$ attacking A_L. So, (by Definition 53) there exists a DT_{A_L} that has just one move, $move_1 = A_L$, DT_{A_L} is succeeded, and $size(DT_{A_L}) = 1$.

Induction: $\Rightarrow$ Assume that $A_L \in F_P^{\uparrow n+1}$, i.e. A_L is $acceptable_{p,o}$ w.r.t. $F_P^{\uparrow n}$. Then for every $A^o_{L'}$ attacking A_L there is a $justified^{p,o}_P$ argument $A^p_{L''}$ in $F_P^{\uparrow n}$

attacking $A^o_{L'}$. By induction hypothesis, all such $A^p_{L''}$ have at least a successful dialogue tree $DT_{A_{L''}}$ with $size(DT_{A_{L''}}) \leq n$. So there is a successful DT_{A_L} with $size(DT_{A_L}) \leq n+1$.

$\Leftarrow$ Assume now that there is a successful dialogue tree DT_{A_L} with $size(DT_{A_L}) \leq n+1$. This means that each argument that attacks A_L is attacked by some $A^p_{L''}$, each one with a successful $DT_{A_{L''}}$ with $size(DT_{A_{L''}}) \leq n$. By induction hypothesis, all such $A^p_{L''}$ belong to $F_P^{\uparrow n}$. So A_L is $acceptable_{p,o}$ w.r.t. $F_P^{\uparrow n}$, i.e. $A_L \in F_P^{\uparrow n+1}$.

Clearly, where the fixpoint of F_P is obtained in an enumerable number of interactions, (1) implies that $A_L \in lfp(F_P)$ iff there exists a successful DT_{A_L}.

Justified arguments for $\perp$

The proof for $justified^{p,o}_P$ arguments for $\perp$ also follows by induction on the size of the tree for $A_\perp$ and the number of iterations of F_P starting from $\emptyset$, needed to obtain $A_\perp$.

Formally, we prove by induction on n that:

(1) $A_\perp \in F_P^{\uparrow n}$ $(n > 1)$ iff there exists a successful $DT_{A_\perp}$ such that

$$size(DT_{A_\perp}) \leq n$$

Base: Let S be the result of $F_P^{\uparrow 0}$. Assume that there are no default literals *not* L in $A_\perp$, i.e. $Assump(A_\perp) = \emptyset$. $A_\perp \in F_P(S)$ iff all objective literals $L_i \in Body(A_\perp)$ $(i > 0)$ are in S, i.e. seen that all $L_i \in Body(A_\perp)$ are $acceptable_{p,o}$ w.r.t. $F_P^{\uparrow n}$ and as $Assump(A_\perp) = \emptyset$ then $A_\perp$ is $acceptable_{p,o}$ w.r.t. $F_P^{\uparrow 1}$.

Induction: $\Rightarrow$ Assume that $A_\perp \in F_P^{\uparrow n+1}$, i.e. $A_\perp$ is $acceptable_{p,o}$ w.r.t. $F_P^{\uparrow n}$. Then for every argument A^o_L attacking $A_\perp$, there is a $justified^{p,o}_P$ argument $A^p_{L'}$ in $F_P^{\uparrow n}$ attacking A^o_L. By induction hypothesis, all such $A^p_{L'}$ have at least a successful dialogue tree $DT_{A_{L'}}$ with $size(DT_{A_{L'}}) \leq n$. So there is a successful $DT_{A_\perp}$ with $size(DT_{A_\perp}) \leq n+1$.

$\Leftarrow$ Assume now that there is a successful $DT_{A_\perp}$ with $size(DT_{A_\perp}) \leq n+1$. This means that each argument that attacks $A_\perp$ is attacked by some $A^p_{L'}$, each one with a successful $DT_{A_{L'}}$ and $size(DT_{A_{L'}}) \leq n$. By induction hypothesis, all such $A^p_{L'}$ belong to $F_P^{\uparrow n}$. So $A_\perp$ is $acceptable_{p,o}$ w.r.t. $F_P^{\uparrow n}$, i.e. $A_\perp \in F_P^{\uparrow n+1}$.

Clearly, where the fixpoint of F_P is obtained in an enumerable number of iterations, (1) implies that $A_\perp \in lfp(F_P)$ iff there exists a successful $DT_{A_\perp}$.

Overruled arguments for L

By Def. 42, there is a $\text{justified}^{o,p}_P$ argument $A^o_{L'}$ attacking A_L iff A_L is $\text{overruled}^{p,o}_P$. As shown above in the proof for $\text{justified}^{p,o}_P$ arguments for L', for each such $\text{justified}^{p,o}_P$ argument $A_{L'}$ there is a successful $DT_{A_{L'}}$. So, if in the failed DT_{A_L}, one of the moves below A_L (i.e. one of the arguments attacking A_L) has a successful dialogue tree (i.e. it is $\text{justified}^{o,p}_P$), then A_L is $\text{overruled}^{o,p}_P$ *q.e.d.* ■

The following two examples illustrate the concepts presented in Proposition 40.

Example 23 *Let* $P1 = \{p \leftarrow not\ \ a;\ a \leftarrow not\ \ b, not\ \ c;\ b \leftarrow not\ \ c;\ c \leftarrow not\ \ g;\ g;\ m \leftarrow not\ \ l;\ l \leftarrow not\ \ m\}$. *Figure 3.12 presents some of the possible dialogue trees in* $P1$.

The last move of $DT^{s,s}_{A_p}$ *(in the centre right of the figure) is the p-argument* A^s_g. *Since* $DT^{s,s}_{A_g}$ *succeeds,* A^s_g *is* $justified^{s,s}_{P1}$. *Therefore,* $DT^{s,s}_{A_c}$ *does not succeed because it is attacked by the o-argument* A^s_g. *Thus,* A^s_p *is* $justified^{s,s}_{P1}$ *and* A^s_c *is* $overruled^{s,s}_{P1}$. *Note that* A^s_m *and* A^s_l *are both* $defensible^{s,s}_{P1}$ *because they are attacked by a* $non\text{-}justified^{s,s}_{P1}$ *argument (in this case because they attack each other).*

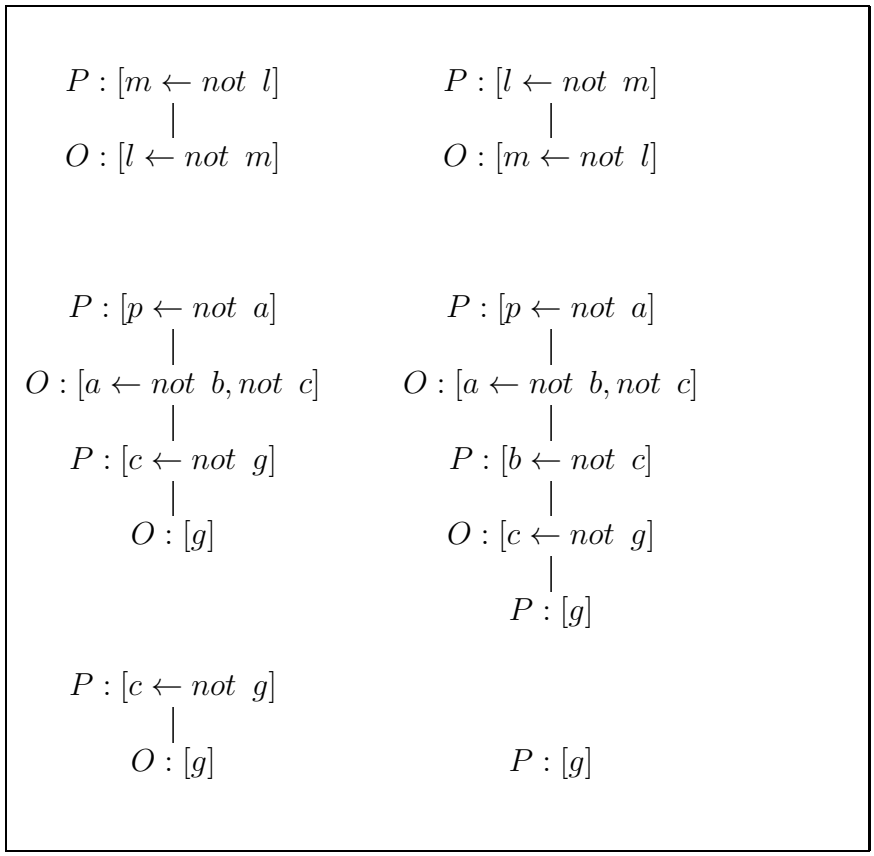


Figure 3.12: Some $DT^{s,s}_{A_L}$ in $\{p \leftarrow not\ \ a;\ a \leftarrow not\ \ b, not\ \ c;\ b \leftarrow not\ \ c;\ c \leftarrow not\ \ g;\ g;\ m \leftarrow not\ \ l;\ l \leftarrow not\ \ m\}$

Example 24 *Let* $P2 = \{a \leftarrow not\ b;\ \neg a;\ b;\ \neg b;\ c;\ \bot \leftarrow c\}$. *Figure 3.13 illustrates the possible* $DT^{w,w}_{A_L}$ *in* $P2$. *Note that each dialogue tree does not succeed because its last move is an o-argument. Nevertheless, all arguments are* $defensible^{w,w}_{P2}$ *because none of these last moves are* $justified^{w,w}_{P2}$.

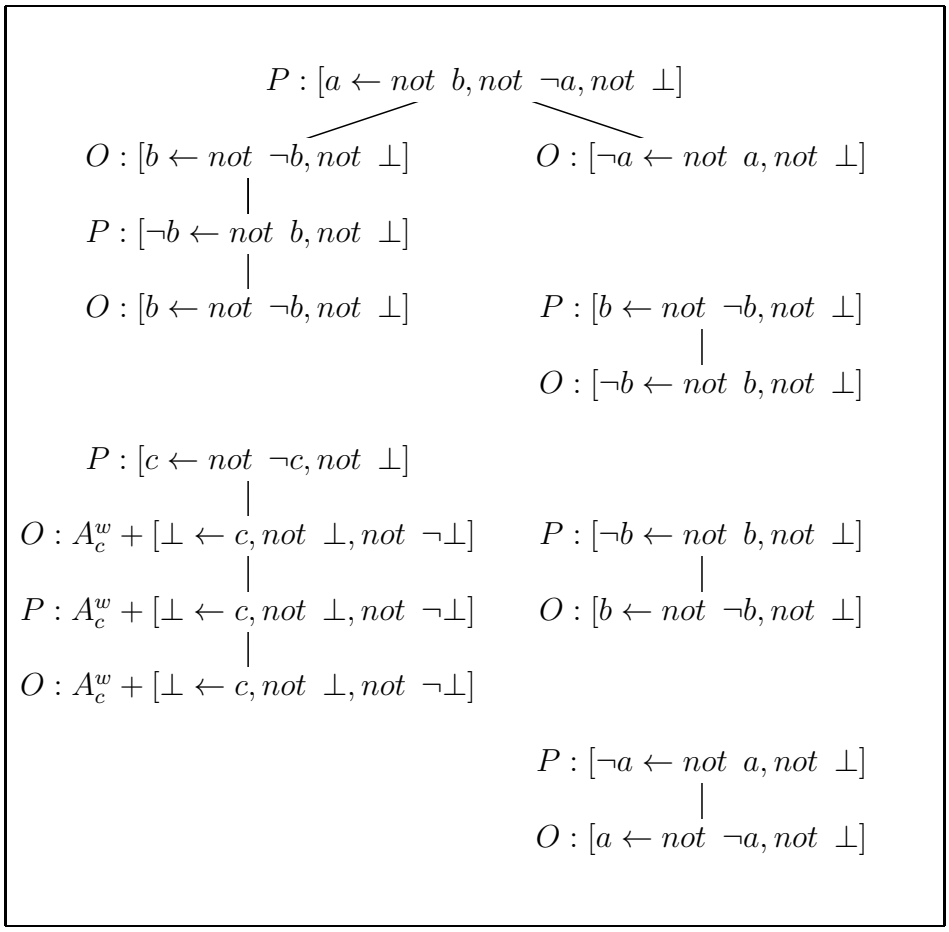


Figure 3.13: $DT^{w,w}_{A_L}$ in $\{a \leftarrow not\ \ b;\ \neg a;\ \ b;\ \neg b;\ \ c;\ \ \bot \leftarrow c\}$

Figure 3.14 illustrates the possible $DT^{s,w}_{A_L}$ *in P2. In this case, all arguments are justified*$^{s,w}_{P2}$.

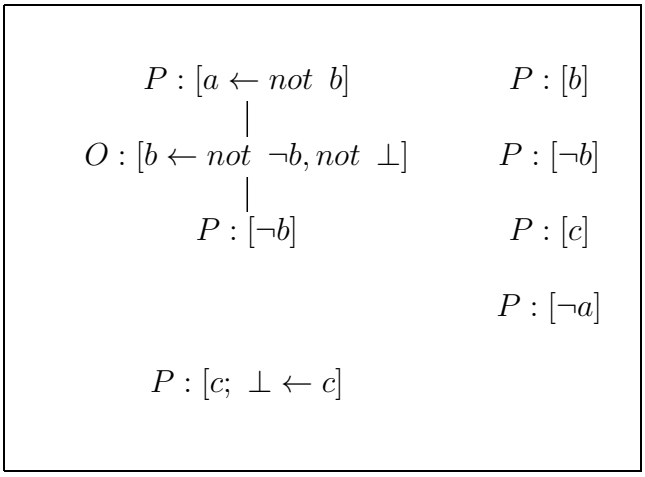


Figure 3.14: $DT^{s,w}_{A_L}$ in $\{a \leftarrow not\ b;\ \neg a;\ b;\ \neg b;\ c;\ \bot \leftarrow c\}$

For justified$^{w,w}_P$ arguments there are no contradiction. But for justified$^{s,k}_P$ arguments there migth be. As in the previous section, here also it is easy to detect the status of a literal w.r.t. contradiction for the case of justified$^{s,w}_P$ literals:

Proposition 41 *A justified*$^{s,w}_P$ *argument* A^s_L *in P is either*

- contradictory$^{s,w}_P$ *iff L is the symbol* $\bot$*, or different from* $\bot$ *and there exists at least a successful* $DT^{s,w}_{A_{\neg L}}$*;*
- based-on-contradiction$^{s,w}_P$ *iff* A^s_L *is not contradictory*$^{s,w}_P$ *and*
  - *there exists a contradictory*$^{s,w}_P$ $A^s_{L'}$ *with a rule* $L' \leftarrow \ldots, L, \ldots$*, or*
  - *there exists an* $L' \in Conc(A^s_L)$ *such that* $A^s_{L'}$ *is contradictory*$^{s,w}_P$*, or*
  - *for all dialogue*$^{s,w}_{A_L}$ *in a successful* $DT^{s,w}_{A_L}$*: the argument of the last move is not non-contradictory*$^{s,w}_P$*; or*
- non-contradictory$^{s,w}_P$*, otherwise.*

Proof. Follows directly from Proposition 40, Definition 48, and Proposition 29 ■

To conclude about the truth value of an objective literal L we evaluate more than one dialogue tree of each argument for such L:

Proposition 42 *An objective literal L is*

- $true_P^{p,o}$ *iff there exists a successful* $DT_{A_L}^{p,o}$. *Thus,* L *is*
  - $contradictory_P^{s,w}$ *iff for all successful* $DT_{A_L}^{s,w}$: A_L^s *is* $contradictory_P^{s,w}$, *or*
  - $based\text{-}on\text{-}contradiction_P^{s,w}$ *iff for all successful* $DT_{A_L}^{s,w}$: A_L^s *is* $based\text{-}on\text{-}contradicton_P^{s,w}$, *or*
  - $non\text{-}contradictory_P^{s,w}$ *iff there exists a successful* $DT_{A_L}^{s,w}$ *such that* A_L^s *is* $non\text{-}contradictory_P^{s,w}$;
- $false_P^{p,o}$ *iff* $\forall$ $DT_{A_L}^{p,o}$: A_L^p *is* $overruled_P^{p,o}$;
- $undefined_P^{p,o}$ *iff* $\forall$ $DT_{A_L}^{p,o}$: A_L^p *is neither* $justified_P^{p,o}$ *nor* $overruled_P^{p,o}$.

Proof. Follows easily from the results above in the section ■

Example 25 *The truth value of literals of* $P1$ *from Example 23 is as follows:* g *and* p *are* $true_{P1}^{s,s}$, c *is* $false_{P1}^{s,s}$, *and* m *and* l *are both* $undefined_{P1}^{s,s}$. *Following Example 24, all literals of* $P2$ *are* $justified_{P2}^{s,w}$. *However, all literals of* $P2$ *are* $undefined_{P2}^{w,w}$.

3.4 On the implementation of the proposed semantics

For this proposal we have made two implementations, both in the XSB System (by resorting to tabling) [SW07]. One is a bottom-up implementation of the semantics, following closely its declarative definition, and the other implements query-driven proof procedure for the semantics. The proof procedure has also been implemented by using the toolkit Interprolog [Cal04], a middleware for Java and Prolog which provides method/predicate calling between both. Since we also describe an Argumentation-based Negotiation System, more complex than those implementation, we prefer to explain both XSB and Interprolog in the next Chapter.

The procedure can be viewed as building a *dialogue tree* in which a proponent and an opponent exchange arguments. In its simplest form, each dialogue in such a dialogue tree is viewed as a sequence of alternating arguments: $P_1, O_2, P_3, \ldots, P_i, O_{i+1}, P_{i+2}, \ldots$. The proponent puts forward some initial argument P_1. For every argument P_i put forward by the proponent, the opponent attempts to respond with an attacking argument O_{i+1} against P_i. For every attacking argument O_i, put forward by the opponent, the proponent attempts to counter-attack with a proposing argument P_{i+1}. For the proponent to win a dialogue, the dialogue sequence has to end with a proposing argument. To win a

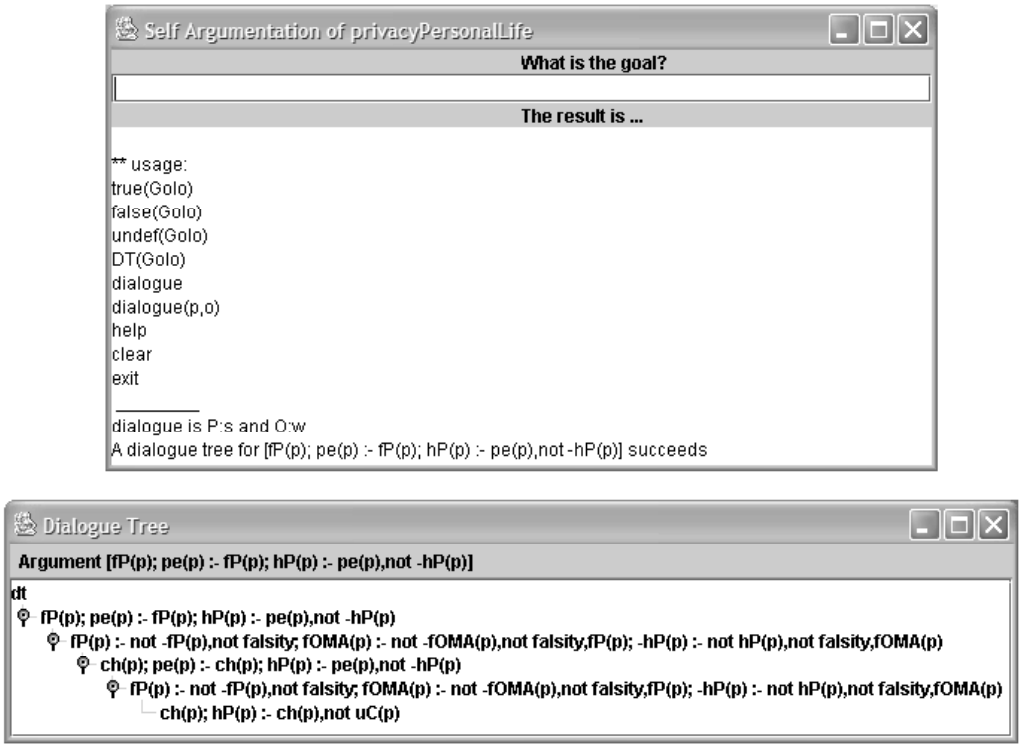


Figure 3.15: A Dialogue Tree $DT^{s,w}_{hp(P)}$ from Example 9

dialogue tree, the proponent must win every dialogue on it. The operational semantics has been implemented by also using the toolkit Interprolog. Figure 3.15 presents a dialogue tree $DT^{s,w}_{PPL}$ for $hP(p)$.

Chapter 4

A Proposal for Argumentation-Based Negotiation

This chapter presents the main contribution of the dissertation: an argumentation-based negotiation semantics for distributed knowledge bases represented as extended logic programs. Such a semantics extends the argumentation semantics presented in the previous chapter by considering sets of (distributed) logic programs, rather than single ones. For specifying the ways in which the various logic programs may combine their knowledge we make use of concepts that have been developed in the areas of defeasible reasoning and multi-agent settings. In particular, we associate to each program P a cooperation set (the set of programs that can be used to complete the knowledge in P) and an argumentation set (the set of programs with which P has to reach a consensus). In this chapter, we first define a declarative semantics for argumentation-based negotiation. Then, some illustrative examples are presented. Finally, we present a general architecture for implementing the semantics.

In this chapter we propose an argumentation-based negotiation semantics for sets of knowledge bases distributed through a multi-agent setting (MAS). In it different agents may have independent or overlapping knowledge bases Kb, each Kb being represented by an *extended logic program with denials* (ELPd) over a *language* (cf. Def. 3). If all agents have complete access to the knowledge bases of all other agents, then they should be able to build arguments using rules from others (i.e.

to cooperate) and would have to defend their arguments against arguments build by others (i.e. to argue). In this case, the semantics of argumentation-based negotiation framework should coincide with the semantics of the union of the knowledge bases, viewed as a single one. Here we want to deal with cases where the semantics of multi-agent setting does not necessarily coincide with the union. The basis of our proposal is that agents negotiate by exchanging parts of their knowledge to obtain a consensus concerning the inference of an agent's beliefs.

Moreover, a multi-agent setting $\mathcal{A}$ might have an agent's knowledge base physically distributed over a computer network. Thus, an agent Ag of $\mathcal{A}$ does not need to, and sometimes cannot, argue and/or cooperate with all agents in $\mathcal{A}$. In our proposal, every agent Ag in $\mathcal{A}$ has associated two sets of agents: the set of agents with which it can cooperate in order to build arguments, and the set of agents from whose attacks it must defend itself (i.e. argue) in order to reach some consensus. In general, little is assumed about these sets: we only impose that every agent argues and cooperates with itself because it would make little sense for an agent neither to access its own knowledge nor to obtain a consensus based upon its own knowledge.

The ability of associating these sets to each agent provides a flexible framework which, besides reflecting the possibly existing physical network, may serve for other purposes than the ones above:

- For modelling knowledge over a hierarchy where each node of the hierarchy is represented by a Kb that cooperates with all its inferiors, and must argue with all its superiors.

- For modelling knowledge that evolves. Here the "present" can use knowledge from the "past" unless this knowledge from the past is in conflict with later knowledge. This can be modelled by allowing any present node to cooperate with its past nodes, and forcing any past node to argue with future nodes.

In these cases, it is important that the knowledge is not flattened, as in the union of all knowledge bases, and that the semantics is parametric on the specific Kb. I.e. it might happen that an argument is acceptable in a given (agent$_i$) Kb_i, and not acceptable in another (agent$_j$) Kb_j of the same system.

As with other argumentation based frameworks (e.g. [Dun95, PS97, BDKT97, Vre97, SS02b, Pol01, GS04]) the semantics is defined based on a notion of acceptable arguments, this notion being itself based on an attacking relation among arguments. Moreover, as in Chapter 3, based on these acceptability, all arguments are assigned a status: justified argument are those that are always acceptable; overruled arguments are those that are attacked by a justified argument; other arguments (which may or may not be acceptable but which are not attacked by a justified one) are called defensible.

It is also a goal of the proposed framework to be able to deal with mutually inconsistent, and even inconsistent, knowledge bases. Moreover, when in presence of contradiction, we want to obtain ways of multi-agent setting reasoning, ranging from consistent (in which inconsistencies lead to no result) to paraconsistent. For achieving this, the agents may exchange strong or weak arguments, as it is made clear in the following. This also yields a refinement of the possible status of arguments: justified arguments may now be contradictory, based on contradiction or non-contradictory.

In the remainder of this chapter, we first present what it is necessary to extend the Self-argumentation semantics from Chapter 3 to this Argumentation-based Negotiation proposal. Some definitions from the former will have to be reset from a centralized point of view to a distributed one. We then illustrate our ideas/proposal with a multi-agent setting that models a trial which clearly represents the argumentative dependency between a prosecuting lawyer (the prosecution) and a defending lawyer (the defense), and a cooperative dependency between a lawyer and her/his witness. After that we define the declarative semantics of the argumentation-based negotiation framework. Then we show some properties of the framework, namely properties that relate it to extant work on argumentation and on other semantics for logic programs. Finally, a couple of illustrative examples.

4.1 From Centralized to Distributed Argumentation

In the centralized proposal[1], a (strong or weak) *argument* of Ag for some objective literal L is a *complete well-defined sequence* for L over the *set of rules* of Ag's knowledge base (cf. Def. 37, Def. 36 and Def. 35, respectively). Since we are concerned with modelling knowledge bases distributed through several agents, (strong and weak) *partial arguments* of Ag for L will also considered in the distributed proposal. By partial argument of Ag for L we mean a non-complete well-defined sequence for L, called Seq_L, over the set of rules of Ag's knowledge base. It occurs if there is no rule for at least one objective literal L_i in the body of some rule in Seq_L. Therefore, even if an agent Ag has a rule for L in its knowledge base, a complete well-defined sequence for L may not be built by Ag alone, and so Ag has only a partial argument for L. The (complete or partial) arguments of Ag built only on its own knowledge base are called *local arguments* of Ag. Since we want to deal with local partial arguments, an argumentation-based semantics with coop-

[1]For simplicity, in the remainder of this chapter we will say "centralized proposal" instead of "self-argumentation proposal" and "distributed proposal" instead of "argumentation-based negotiation proposal".

eration is proposed. By *argumentation*, we mean the evaluation of arguments to obtain a consensus about common knowledge; by *cooperation*, we mean obtaining arguments to achieve knowledge completeness.

Intuitively, the distributed proposal coincides with the centralized one if every agent argues and cooperates with all agents in a multi-agent setting (MAS). As has already been said, we want to deal with cases where these proposals do not coincide, i.e. when an MAS represents a kind of hierarchy of knowledge. Moreover, the MAS might have the agent's knowledge base physically distributed in a computer network. Therefore, an agent Ag does not need to argue and/or to cooperate with all agents in an MAS. We associate with every agent in an MAS an unique identifier α and two sets of agents' identifiers, corresponding to its argumentative and cooperative agents (the former is denoted $\mathcal{A}rgue_\alpha$ and the latter $\mathcal{C}ooperate_\alpha$). We further impose that every agent argues and cooperates with itself because it would make little sense for an agent neither to access its own knowledge nor to obtain a consensus based upon its own knowledge. Local partial arguments of agent Ag can be completed with arguments from $\mathcal{C}ooperate_\alpha$, and completed arguments[2] of Ag are evaluated with respect to arguments from $\mathcal{A}rgue_\alpha$. Furthermore, the evaluation of Ag's arguments is only with respect to arguments from agents from both $\mathcal{A}rgue_\alpha$ and $\mathcal{C}ooperate_\alpha$.

Remark 43 *For simplicity, we will say "(a local partial argument is completed) via cooperation with $\mathcal{C}ooperate_\alpha$" instead of "(a local partial argument is completed) given a set of arguments from agents in $\mathcal{C}ooperate_\alpha$" and "an argument is evaluated by $\mathcal{A}rgue_\alpha$" instead of "an argument is evaluated with respect to arguments from agents in $\mathcal{A}rgue_\alpha$".*

A multi-agent setting is seen as a set of agents such that each agent Ag is a tuple

$$< \alpha, Kb_\alpha, \mathcal{A}rgue_\alpha, \mathcal{C}ooperate_\alpha >$$

with Ag's identity in the MAS (denoted by $Id(Ag)$), an extended logic program with denials which represents the Ag's knowledge base (denoted by $Kb_{Id(Ag)}$), and the sets of argumentative and cooperative agents (denoted by $\mathcal{A}rgue_{Id(Ag)}$ and $\mathcal{C}ooperate_{Id(Ag)}$, respectively). For instance, let $\mathcal{A}$ be a multi-agent setting with three agents: Ag_1, Ag_2, and Ag_3, such that Ag_1 argues with Ag_2 and Ag_3, and both Ag_1 and Ag_2 cooperate with Ag_3. The multi-agent setting for the above description is written as follows:

$$\begin{array}{rl} \mathcal{A} = \{ & Ag_1 =< 1, Kb_1, \{1,2,3\}, \{1,3\} >, \\ & Ag_2 =< 2, Kb_2, \{2\}, \{2,3\} >, \\ & Ag_3 =< 3, Kb_3, \{3\}, \{3\} > \ \} \end{array}$$

[2]Here, we also consider local partial arguments that have been completed through cooperation.

where 1 (resp.2 and 3) is the identity of agent Ag_1 (resp.Ag_2 and Ag_3). For faster understanding, the argumentation and cooperation relations are depicted in a directed graph such as the one in Figure 4.1, representing agents Ag_1, Ag_2, and Ag_3 from $\mathcal{A}$.

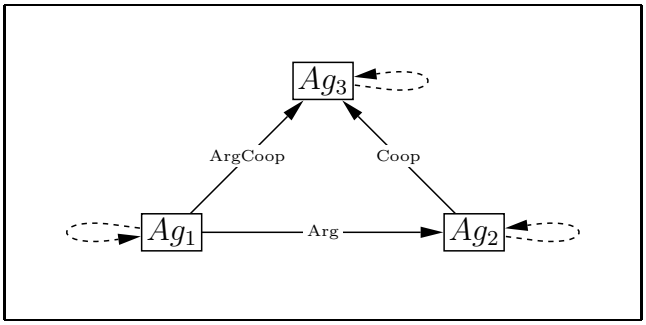


Figure 4.1: An example of a Multi-agent Setting

Remark 44 *[Directed graph illustrating an MAS] Each node in a directed graph illustrating an MAS represents an agent Ag. A dashed loop represents the fact that an agent argues and cooperates with itself. Furthermore, a directed arc with label Arg, Coop, or ArgCoop links an agent to its argumentative, cooperative, and argumentative/cooperative agents, respectively. A link Coop (resp. Arg) from Ag_1 to Ag_2 means that Ag_1 asks for cooperation (resp. to argue) with Ag_2.*

The centralized proposal defines the status of an argument as justified, overruled, or defensible based on its acceptability (see Def. 44). Moreover, a justified argument can be attacked by the agent's arguments, but such arguments are either overruled or defensible (i.e. "non-justified" arguments). This makes sense in such a proposal, because it considers a knowledge base Kb of just one agent Ag, and so any argument over Kb is evaluated only by Ag itself. In the distributed proposal, arguments of an agent Ag can be built by Ag alone or via cooperation with $\mathcal{C}ooperate_\alpha$. However, it would make little sense for an agent to cooperate with another one by giving the latter arguments that are overruled in the former.

We thus impose the idea that every argument Arg of an agent Ag can be used in a cooperation process if and only if Arg is initially evaluated by $\mathcal{A}rgue_\alpha$. However, we consider that $\mathcal{A}rgue_\alpha$ can evaluate Arg as defensible, and such an argument might be evaluated as justified by other set of argumentative agents. In other words, Ag has a defensible argument Arg with respect to $\mathcal{A}rgue_\alpha$, Arg could be used by another agent Ag' (by considering that $\alpha \in \mathcal{C}ooperate_{\alpha'}$) and so this argument might be justified with respect to $\mathcal{A}rgue_{\alpha'}$. Therefore, an agent can cooperate with both justified and defensible arguments.

As presented in the centralized proposal, the truth value of an agent's belief could be true (and either contradictory, based on contradiction, or non-contradictory), false, or undefined (see both Definition 49 and Proposition 30). However, in the distributed proposal, an agent's belief should only be deduced with respect to both sets of argumentative and cooperative agents. Intuitively, we can deduce that different truth values of a given literal L over a multi-agent setting $\mathcal{A}$ may be obtained. It happens because it depends on which agent the literal L is inferred from, and also on what the specification of both sets of cooperative and argumentative agents is, given all the agents in $\mathcal{A}$. For instance, assume a multi-agent setting where each agent represents "acquired knowledge" in a different period of time, namely *past* and *present*, such that $Kb_{pa} = \{wait \leftarrow not\ \neg train\}$ and $Kb_{pr} = \{\neg train\}$, respectively, are the agents knowledge bases. Intuitively, *present* may ask for cooperation with *past* but not vice-versa. Figure 4.2 illustrates such a description. The literal *wait* is true in *past* because the assumption '$not\ \neg train$' is true. Since *present* asks for cooperation with *past*, the literal *wait* might also be true in *present*. However, such a result is different in *present* because it has new knowledge (i.e. $\neg train$) which causes the assumption '$not\ \neg train$' to be false. Note that, if we want to obtain the same truth value of such a belief in both *past*

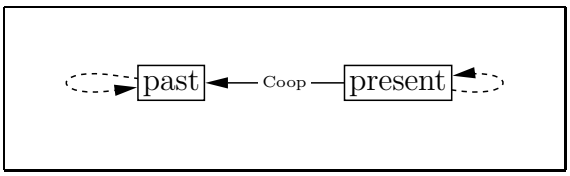


Figure 4.2: $\mathcal{A} = \{< pa, Kb_{pa}, \{pa\}, \{pa\} >, < pr, Kb_{pr}, \{pr\}, \{pr, pa\} >\}$

and *present*, then *present* should argue with *past* (but not vice-versa). This, however, will cause the truth value of any belief in $\mathcal{A}$ to correspond to its truth value in *present*, i.e. in the "latest period of the time".

Finally, as in the centralized proposal, the distributed proposal may also define ways of multi-agent reasoning ranging from consistent to paraconsistent. Since these two approaches have been fully discussed in the centralized version, we will focus our attention on the details related to distributed knowledge bases.

4.2 "Reaching a Verdict", an example

The following example, *The Inconvenient Witness*, was extracted from a thriller. We use this narrative to model a trial as a multi-agent setting when each agent represents the knowledge of a trial of a different participant: a defending lawyer (the defense), a prosecuting lawyer (the prosecution), and both the witness for the

prosecution (prosecuting witness) and the witness for the defense (defense witness). For the sake of simplicity, we assume the accused is represented by his defense, not being present at the trial.

Example 26 (The Inconvenient Witness) *It was well after 3 a.m. when Emily Brown woke up, frightened. Oh, no. That nasty nightmare again. John Thorn, her lover, was still fast asleep. She looked out of the window at the enchanting full moon. Suddenly she noticed two men, apparently arguing in the garden outside, just a few metres under John's first floor window. From behind the curtains she could clearly see the menacing face of one of them, the one holding a long, shining knife against the elegant overcoat of the other. Suddenly, there was a suffocated scream and the elegant man fell to the ground, stabbed twice. The other quickly opened a car, threw the body in, and sped off down the lane. She could read the registration number. In her head, the twisted face of the victim, a face she already knew, but who? John was still peacefully at rest.*

"It's him!", Emily mumbled, while a sharp shiver went up her spine when, two days later, she saw the photo under the Morning Tribune headline: "Body of Ronald Stump, king of media, found stabbed to death in stone-pit! Chinese suspect identified by Police." But she had seen the killer. He was not Chinese! But how could she explain being an eyewitness. Any testimony would cost her marriage, and her privileged lifestyle. She was faced with a dilemma!

"Tell me, Mr. Thorn", asked Dr. William Watson, the prosecution attorney, "do you sleep with your contact lenses in"? "No", replied Paul. "I sincerely hope you know what perjury means. Do you want us to believe that you put your lenses in, and take them out again, just to quench your thirst when you wake up in the middle of the night?", queried Dr. Watson. An uproar spread in the crowded Wimbledon Court, when the judge brandished his wooden hammer ...

In any trial, the prosecution and the defense argue with each other, and both prosecution and defense witnesses cooperate with, respectively, the prosecution and the defense. Based on *The Inconvenient Witness* description, John is the defense witness because Emily cannot be in Court. Emily provides (i.e. cooperates with) John with the necessary statement for his testimony. Therefore, the multi-agent setting

$$\mathcal{T}rial = \{defence, prosecution, policeman, john, emily\}$$

represents the trial described in Example 26. Figure 4.3 illustrates both the arguing and the cooperating dependency between the agents.

In the following, we present every agent's knowledge base, as an extended logic program, and the corresponding tuple. Emily saw the murder and knows that the culprit is not a Chinese man. The rules $seen(murder, culprit)$ and $\neg Chinese(culprit)$ model her moral duty to testify.

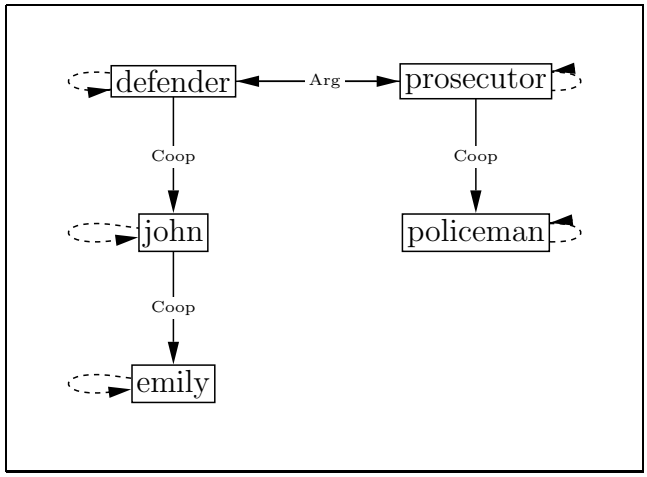


Figure 4.3: "The inconvenient witness"

$$emily =< em, \{se(mu, cu);\ \neg ch(cu)\}, \{em\}, \{em\} >$$

The simplest agent in $\mathcal{T}rial$ is John. He was sleeping when the murder occurred, and so he has no knowledge himself about the crime. However, John is the defense witness and so he needs cooperation from Emily.

$$john =< jo, \emptyset, \{jo\}, \{jo, em\} >$$

The prosecution witness is a policeman who identifies a Chinese man as responsible for the crime. Since the description does not specify how the man was identified, we model such knowledge as $identifiedResponsibleFor(culprit, murder)$.

$$policeman =< po, \{idRF(cu, mu)\}, \{po\}, \{po\} >$$

Both prosecution and defense should have acceptable arguments to support their allegations. Under criminal law, people are presumed innocent until proven guilty of criminal action. To incriminate a culprit, the prosecution should have an allegation of his criminal action and at least one piece of evidence to support it. Evidence makes the culprit seem guilty of a crime. In this case, the evidence is based on the identification of the person assumed to be responsible for the murder[3].

[3]For simplicity, we use P, E, C, and R instead of 'Person', 'Event', 'Culprit', and 'Reason' in the remaining rules in this section. Moreover, we do not make explicit the witness in the rules because both defense and prosecution have only one each. Furthermore, to the right of each rule is the element in the trial which has such 'knowledge'.

$guilty(P, E) \leftarrow allegation(P, E).$ [prosecution]
$allegation(P, E) \leftarrow occurs(E), evidence(P, E).$ [prosecution]
$occurs(murder).$ [prosecution]
$evidence(P, E) \leftarrow identifiedResponsibleFor(P, E), chinese(P).$ [prosecution]
$chinese(culprit).$ [prosecution]

As has already been said, a culprit person is presumed innocent until proven guilty. Since we want to deal with contradictory conclusions, we use $\neg guilty$ instead of *innocent*. So $\neg guilty \leftarrow not\ guilty$ expresses the idea that some person is explicitly not guilty of a crime if there is no evidence that she/he is guilty of it. For simplicity, we consider such a rule as part of the defense's knowledge, i.e. we do not consider it as the prosecution's knowledge.

$$\neg guilty(P, E) \leftarrow not\ \ guilty(P, E). \quad \text{[defence]}$$

Even if there is evidence against her client, the defense may state the absurdity of her client being guilty when the evidence is not admissible. In this case, the inadmissibility is based on a testimony clarifying that the evidence was forged, because the culprit is not a Chinese man.

$\bot \leftarrow guilty(P, E), \neg admissibleEvidence(P, E, R).$ [defence]
$\neg admissibleEvidence(P, E, R) \leftarrow evidence(P, E), forgedEvidence(E, R).$ [defence]
$forgedEvidence(E, C) \leftarrow seen(E, C), \neg chinese(C).$ [defence]

On the other hand, the prosecution may defeat such a testimony by pointing out the absurdity of accepting it, because the witness of the defense forged the evidence and so is committing perjury. As described in Example 26, John has not seen the culprit[4].

$\bot \leftarrow forgedEvidence(E, R), perjury(E, R).$ [prosecution]
$perjury(E, C) \leftarrow \neg seen(E, C).$ [prosecution]
$\neg seen(murder, culprit).$ [prosecution]

Then the prosecution and the defense agents are as follows:

$$\begin{aligned} prosecution = &\ < pr, Kb_{pr}, \{pr, de\}, \{pr, po\} > \\ defence = &\ < de, Kb_{de}, \{de, pr\}, \{de, jo\} > \end{aligned}$$

[4]For simplicity, we do not represent the reasons for this conclusion, and so the prosecution's knowledge about it is a fact.

such that Kb_{pr} and Kb_{de} are

$$Kb_{pr} = \{ \quad gu(P,E) \leftarrow al(P,E);\ al(P,E) \leftarrow oc(E), ev(P,E);\ oc(mu);\ ch(cu);$$
$$ev(P,E) \leftarrow idRF(P,E), ch(P);\ \bot \leftarrow fE(E,R), pe(E,R);$$
$$pe(E,C) \leftarrow \neg se(E,C);\ \neg se(mu,cu) \ \}$$

$$Kb_{de} = \{ \quad \neg gu(P,E) \leftarrow not\ gu(P,E);\ \bot \leftarrow gu(P,E), \neg aE(P,E,R);$$
$$\neg aE(P,E,R) \leftarrow ev(P,E), fE(E,R);\ fE(E,C) \leftarrow se(E,C), \neg ch(C) \ \}$$

4.3 Declarative Semantics

As motivated in the introduction, in our framework the knowledge bases of agents are modelled by logic programs. More precisely, we use *Extended Logic Programs with denials* over a *l*anguage $\mathcal{L}$ (cf. Def. 3 and Def. 1, respectively), themselves an extension of Extended Logic Programs [GL90], for modelling the knowledge bases.

As already motivated in the introduction we propose an argumentation-based semantics with cooperation. By *argumentation*, we mean the evaluation of arguments to obtain a consensus about common knowledge, and by *cooperation*, we mean the granting of arguments to achieve knowledge completeness. Then, besides the knowledge base, in our framework each argumentative agent Ag in a multi-agent setting $\mathcal{A}$ must have a unique identity α in $\mathcal{A}$, and two sets of agents' identifiers corresponding to argumentative and cooperative agents with Ag. Moreover, the identity of Ag is in both sets of argumentative and cooperative agents with Ag:

Definition 54 (Argumentative Agent) *An* argumentative agent *(or* agent*, for short) over a language $\mathcal{L}$ and a set of identifiers Ids is a tuple*

$$Ag =< \alpha,\ Kb_\alpha,\ \mathcal{A}rgue_\alpha,\ \mathcal{C}ooperate_\alpha >$$

where $\alpha \in Ids$, Kb_α is an ELPd over $\mathcal{L}$, $\mathcal{A}rgue_\alpha \subseteq Ids$ and $\mathcal{C}ooperate_\alpha \subseteq Ids$ such that $\alpha \in \mathcal{A}rgue_\alpha$ and $\alpha \in \mathcal{C}ooperate_\alpha$.

We denote by $Id(Ag)$ (resp. $Kb_{Id(Ag)}$, $\mathcal{A}rgue_{Id(Ag)}$ and $\mathcal{C}ooperate_{Id(Ag)}$), the 1st (resp. 2nd, 3rd, and 4th) position of the tuple Ag, and by $\mathcal{H}(Id(Ag))$ (cf. Def. 1) the Extended Herbrand Base of $Kb_{Id(Ag)}$.

Hereafter, we say 'arguments from $\mathcal{C}ooperate_{Id(Ag)}$ (or $\mathcal{A}rgue_{Id(Ag)}$)' instead of 'arguments from agents whose identities are in $\mathcal{C}ooperate_{Id(Ag)}$ (or $\mathcal{A}rgue_{Id(Ag)}$)'.

Definition 55 (Multi-agent Argumentative Setting) *Let $\mathcal{L}$ be a language, and Ids be a set of identifiers. A* Multi-Agent argumentative setting *(or* Multi-Agent setting*, for short) $\mathcal{A}$ is a set of agents*

$$\mathcal{A} = \{Ag_1, \ldots, Ag_n\}$$

such that all of the Ag_is are agents over $\mathcal{L}$ and Ids, and no two Ag_is have the same identifier. The Extended Herbrand Base of $\mathcal{A}$, $\mathcal{H}(\mathcal{A})$, is the union of all $\mathcal{H}(\alpha_i)$ such that $\alpha_i \in Ids$.

The (complete or partial) arguments of an agent Ag are built by Ag alone. These are considered *local arguments of Ag*. Definitions of both *local complete argument* and *local partial argument* of an agent Ag for a literal are very similar to the definition of an *argument* of an ELPd for a literal from the centralized proposal (cf. Def. 37). Furthermore, we use the definitions of *Set of Rules* and *Well-defined Sequence* from that proposal (cf. Def. 35 and Def. 36, respectively).

Since we are concerned with modelling knowledge bases distributed over a multi-agent setting, partial arguments of Ag for L must be considered. In fact, an agent alone may not have in its knowledge base enough rules to form a complete argument, but may have part of an argument (a partial argument) that can be completed with knowledge from other agents with which it is able to cooperate. By a partial argument of Ag for L we mean a non-complete well-defined sequence for L, called Seq_L, over the set of rules of Ag's knowledge base. The complete and partial arguments of Ag built only with its own rules are called local arguments of Ag.

Definition 56 (Local (Partial or Complete) Argument) *Let $\mathcal{A}$ be a MAS, Ag be an agent in $\mathcal{A}$, $\alpha = Id(Ag)$, Kb_α be the ELPd of Ag, R^s_α (resp. R^w_α) be the strong (resp. weak) set of rules of Kb_α, and $L \in \mathcal{H}(\mathcal{A})$.*

A strong *(resp.* weak*)* local partial argument of α *for L is a pair (α, Seq_L) such that Seq_L is a well-defined sequence for L over R^s_α (resp. R^w_α). A* strong *(resp.* weak*)* local complete argument of α *for L is any strong (resp. weak) partial local argument (α, Seq_L) such that Seq_L is complete and non-empty. We say that (α, Seq_L) is a* k-local argument of α *for L, $LA^k_\alpha(L)$, if it is either a local partial argument or a local complete argument over R^k_L of α for L (where k is either s, for strong arguments, or w, for weak ones).*

The *set of local arguments* of Ag contains all possible local (complete and partial) arguments over Ag's knowledge base.

Definition 57 (Set of Local Arguments) *Let $\mathcal{A}$ be a MAS, α be an agent's identity in $\mathcal{A}$, and $k \in \{s, w\}$. The set of k-local arguments of α is*

$$LA^k_{\mathcal{A}}(\alpha) = \bigcup_{L \in \mathcal{H}(\mathcal{A})} LA^k_\alpha(L)$$

where $LA^k_\alpha(L)$ is the set of all (strong or weak) local arguments of α for L. The Set of Local Arguments *of α in $\mathcal{A}$ is*

$$LA_{\mathcal{A}}(\alpha) = LA^s_{\mathcal{A}}(\alpha) \cup LA^w_{\mathcal{A}}(\alpha)$$

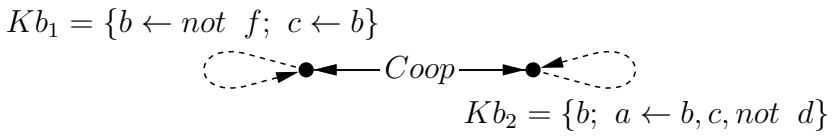
and we denote by $LA(\mathcal{A})$ the union of all local arguments of agents in $\mathcal{A}$.

An agent Ag in a multi-agent setting $\mathcal{A}$ may be able to build a (partial or complete) argument for any objective literal L in $\mathcal{H}(\mathcal{A})$ even though Ag has no knowledge about such an L (i.e. there is no rule $L \leftarrow Body$ in $Kb_{Id(Ag)}$). This may be done through the introduction of an argument with an empty well-defined sequence in the set of local arguments of Ag. From Definition 56, a local partial argument for any $L \in \mathcal{H}(\mathcal{A})$ of Ag is a pair $(Id(Ag), Seq_L)$, such that Seq_L is a well-defined sequence for L over the set S of rules of $Kb_{Id(Ag)}$. If there is no rule for such an L in S, then $Seq_L = []$. Thus, [] is a well-defined sequence for any literal $L \in \mathcal{H}(\mathcal{A})$ and $(Id(Ag), [])$ might be completed with the "help" of some set of (complete and partial) arguments from $\mathcal{C}ooperate_{Id(Ag)}$. Example 27 illustrates how the set of local arguments of Ag is built. After the example, we present the way in which every local partial argument of Ag may be completed.

Example 27 *Let $\mathcal{A} = \{Ag_1, Ag_2\}$ be a MAS such that*

$$\begin{array}{lll} Ag_1 & = < & 1, \{b \leftarrow not\ \ f; c \leftarrow b\},\ \{1\},\ \{1,2\}\ > \\ Ag_2 & = < & 2, \{b; a \leftarrow b, c, not\ \ d\},\ \{2\}, \{1,2\}\ > \end{array}$$

and which is represented by the following directed graph (for details, see Remark 44).



The set of local arguments of agent 1 is

$$\begin{array}{lll} LA_{\mathcal{A}}(1) = \{ & (1, [b \leftarrow not\ \ f]), (1, [b \leftarrow not\ \ f, not\ \ \neg b, not\ \ \bot]), \\ & (1, [c \leftarrow b]), (1, [c \leftarrow b, not\ \ \neg c, not\ \ \bot]), \\ & (1, [b \leftarrow not\ \ f;\ c \leftarrow b]), (1, []), \\ & (1, [b \leftarrow not\ \ f, not\ \ \neg b, not\ \ \bot;\ c \leftarrow b, not\ \ \neg c, not\ \ \bot])\ \} \end{array}$$

and the set of local arguments of agent 2 is

$$\begin{array}{lll} LA_{\mathcal{A}}(2) = \{ & (2, [b]), (2, [b \leftarrow not\ \ \neg b, not\ \ \bot]), \\ & (2, [a \leftarrow b, c, not\ \ d]), (2, [a \leftarrow b, c, not\ \ d, not\ \ \neg a, not\ \ \bot]), \\ & (2, [b;\ a \leftarrow b, c, not\ \ d]), (2, []) \\ & (2, [b \leftarrow not\ \ \neg b, not\ \ \bot;\ a \leftarrow b, c, not\ \ d, not\ \ \neg a, not\ \ \bot])\ \} \end{array}$$

The set of local arguments of $\mathcal{A}$ is $LA(\mathcal{A}) = LA_{\mathcal{A}}(1) \cup LA_{\mathcal{A}}(2)$.

To complete a local partial argument of an agent Ag with (partial or complete) arguments from $\mathcal{C}ooperate_{Id(Ag)}$, we need first to define an *operator* to concatenate these arguments in terms of well-defined sequences.

Definition 58 (+ Operator) *Let $1 \leq i \leq n$, Seq_i be a well-defined sequence for an objective literal L_i, and R_i be the set of all rules in Seq_i.*

The concatenation $Seq_1 + \ldots + Seq_n$ is the set of all well-defined sequences for L_n over $\bigcup_{i=1}^{n} R_i$.

In short, several distinct well-defined sequences are obtained when concatenating two or more well-defined sequences. Furthermore, we can obtain well-defined sequences that are not in fact complete. The operator + comprises all such obtained sequences, as illustrated in the example below:

Example 28 *Following Example 27, Ag_1 has a local complete argument $(1, Seq_c)$ and Ag_2 has a local partial argument $(2, Seq_a)$ such that $Seq_c = [b \leftarrow not\ \ f; c \leftarrow b]$ and $Seq_a = [b; a \leftarrow b, c, not\ \ d]$. By concatenating Seq_c and Seq_a we have the following set of well-defined sequences for the objective literal a over the set of rules*

$$\{b \leftarrow not\ \ f; b; c \leftarrow b; a \leftarrow b, c, not\ \ d\}$$

$$\begin{aligned} S = Seq_c + Seq_a = \ \{\ & [],\ \ [a \leftarrow b, c, not\ \ d],\ \ [b; a \leftarrow b, c, not\ \ d], \\ & [b \leftarrow not\ \ f; a \leftarrow b, c, not\ \ d],\ \ [c \leftarrow b; a \leftarrow b, c, not\ \ d], \\ & [b \leftarrow not\ \ f; c \leftarrow b; a \leftarrow b, c, not\ \ d], \\ & [b; c \leftarrow b; a \leftarrow b, c, not\ \ d]\ \ \} \end{aligned}$$

and so Ag_2 might have two complete local arguments for the literal a (i.e. the last two well-defined sequences for a in S).

We introduce cooperation by defining a *set of available arguments* of an agent Ag given a set S of (complete or partial) arguments. Every (complete or partial) argument of Ag in S is considered an available argument of Ag. Moreover, if a partial argument for an objective literal L of Ag may be further completed with arguments in S belonging to $\mathcal{C}ooperate_{Id(Ag)}$, this further completed argument is also available.

Definition 59 (Set of Available Arguments) *Let $\mathcal{A}$ be a MAS and α be an agent's identity in $\mathcal{A}$. The set of available arguments given a set S of arguments, $Av(S)$, is the least set such that:*

- *if $(\alpha, Seq_L) \in S$ then $(\alpha, Seq_L) \in Av(S)$, and*
- *if $\exists\{(\beta_1, Seq_{L'}), \ldots, (\beta_i, Seq_L)\} \subseteq Av(S)$ and $\{\beta_1, \ldots, \beta_i\} \subseteq \mathcal{C}ooperate_{\alpha}$ then for any $NSeq_L \in Seq_{L'} + \ldots + Seq_L$, $(\alpha, NSeq_L) \in Av(S)$*

where $\beta_1, \ldots, \beta_i$ *are agent's identifiers in* $\mathcal{A}$. *Let* $LA(\mathcal{A})$ *be the set of local arguments of* $\mathcal{A}$. *We denote by* $\mathcal{A}rgs(\mathcal{A})$ *the* set of available arguments of $\mathcal{A}$ *given* $LA(\mathcal{A})$, *and dub it the* set of all available arguments in $\mathcal{A}$. *Members of* $\mathcal{A}rgs(\mathcal{A})$ *will, as usual, be called* arguments.

As in the centralized proposal, also here we are concerned with obtaining ways of reasoning, ranging from consistent to paraconsistent. For this, the agents cooperate and argue by exchanging strong and weak arguments. Similarly to the centralized proposal, we assume that every proponent (resp. opponent) agent in a given multi-agent setting exchanges arguments in the same way, i.e. every proposed (resp. opposing) argument is a strong or weak argument. The following two propositions reinforce such an assumption. According to the first proposition, every available argument is of the same kind, strong or weak, as the given set of arguments. From the second proposition, we see that an agent might have all arguments from its cooperative agents.

Proposition 45 *If* S *is a set of strong (resp. weak) arguments, then* $Av(S)$ *is also a set of strong (resp. weak) arguments*

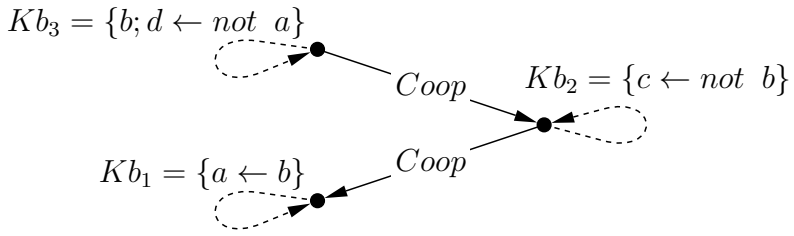
Proof. Trivial, since S contains only strong (resp. weak) arguments, we only have a set of strong (resp. weak) rules and so we only build strong (resp. weak) arguments ■

Proposition 46 *Any available argument* (β, Seq_L) *of* β *for* L *is an available argument* (α, Seq_L) *of* α *for* L *if* $\beta \in \mathcal{C}ooperate_\alpha$

Proof. By Definition 57, $(\alpha, [])$ is an available argument of α for any literal in $\mathcal{H}(\mathcal{B})$. Assume (β, Seq_L) is an available argument of β such that $\beta \in \mathcal{C}ooperate_\alpha$. By Definition 58, we have $Seq_L \in [] + Seq_L$ and so we obtain the available argument (α, Seq_L) of α for L ■

The following example illustrates how available arguments are built via operator $+$. This example also depicts available arguments built by agents that are not directly interrelated, i.e. there is an "indirect cooperation" between such agents.

Example 29 *Consider the following graph representing a multi-agent setting* $\mathcal{A}$. *Note that agent* 1 *may use arguments from agent* 3 *because* $3 \in \mathcal{C}ooperate_2$ *and* $2 \in \mathcal{C}ooperate_1$.



The set of strong local arguments of $\mathcal{A}$ *is*

$$LA^s(\mathcal{A}) = \{(1,[]),(1,[a \leftarrow b]),(2,[]),(2,[c \leftarrow not\ b]),(3,[]),(3,[b]),(3,[d \leftarrow not\ a])\}$$

For simplicity, we call $LA^s(\mathcal{A})$ *as* S*. Based on the first condition of Def. 59, every argument in* S *is an available argument, i.e.* $S \subseteq Av(S)$*. Based on the second condition of Def. 59, we further obtain the following available arguments:*

- $(1,[c \leftarrow not\ b])$ *because* $\{(2,[c \leftarrow not\ b]),(1,[])\} \subset Av(S)$*;*
- *since* $\{(3,[b]),(2,[])\} \subset Av(S)$*,* $(2,[b]) \in Av(S)$*; similarly* $(2,[d \leftarrow not\ a]) \in Av(S)$*, because* $\{(3,[d \leftarrow not\ a]),(2,[])\} \subset Av(S)$*;*
- *consequently,* $(1,[b])$ *and* $(1,[d \leftarrow not\ a])$ *are available arguments because* $\{(2,[b]),(1,[])\} \subset Av(S)$ *and* $\{(2,[d \leftarrow not\ a]),(1,[])\} \subset Av(S)$*, respectively;*
- *because* $\{(1,[a \leftarrow b]),(1,[b])\} \in Av(S)$*,* $(1,[b;\ a \leftarrow b]) \in Av(S)$ [5]*.*

The least set of available arguments of $\mathcal{A}$ *given* $LA^s(\mathcal{A})$ *is*

$$\begin{aligned} Av(LA^s(\mathcal{A})) = \ & LA^s(\mathcal{A}) \cup \{\ (1,[c \leftarrow not\ b]),\ (2,[b]),\ (2,[d \leftarrow not\ a]), \\ & (1,[b]),\ (1,[d \leftarrow not\ a]),\ (1,[b; a \leftarrow b])\ \} \end{aligned}$$

From Definition 59, $Args(\mathcal{A})$ contains all available arguments of an agent Ag built via cooperation with agents in $\mathcal{C}ooperate_{Id(Ag)}$. However, some of these arguments might not be acceptable with respect to arguments in $Args(\mathcal{A})$ from $\mathcal{A}rgue_{Id(Ag)}$. We now describe how a negotiation process should be performed, where cooperation and argumentation are interleaved processes. Initially, assume that an available argument A of Ag is acceptable w.r.t. a set of arguments S if every argument against A from $\mathcal{A}rgue_{Id(Ag)}$ is attacked by an argument in S. Intuitively, if an agent Ag builds an available argument A by concatenating its local partial argument with arguments from $\mathcal{C}ooperate_{Id(Ag)}$, then A must be evaluated

[5] $(1,[b;\ a \leftarrow b])$ can also be obtained from the set of available arguments $\{(1,[a \leftarrow b]),(2,[b])\}$. Both ways to complete the local partial argument $(1,[a \leftarrow b])$ of agent Ag_1 are correct.

by every argumentative agent in $\mathcal{A}rgue_{Id(Ag)}$. It means that each argumentative agent Ag^a should try to build an available argument CA against A. Two situations may occur:

1. Ag^a argues and cooperates only with itself. If Ag^a cannot build a complete argument CA by itself, and since there is no other agent to cooperate with Ag^a, Ag^a cannot argue against A. On the other hand, if CA is built by Ag^a, Ag^a does not need evaluation of CA by any agent other than itself, and so Ag^a may use its argument against A; or

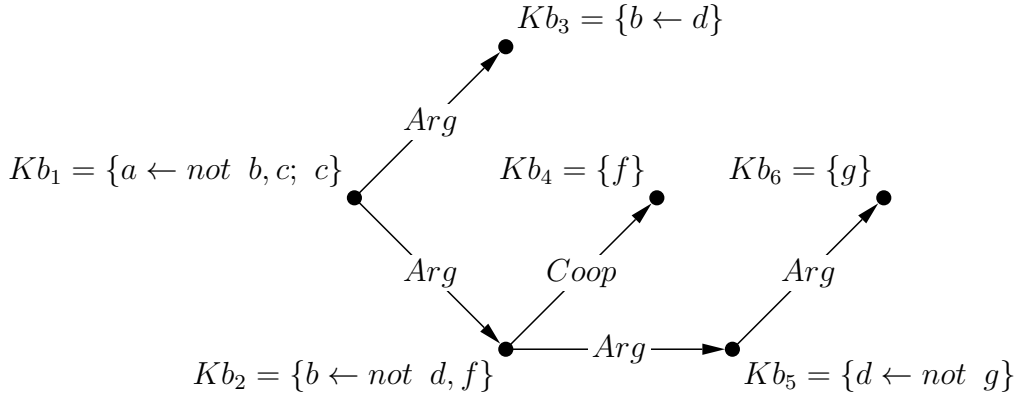
2. Ag^a argues and/or cooperates with other agents. In such a case, building CA may require the concatenation of arguments from $\mathcal{C}ooperate_{Id(Ag^a)}$ and then the evaluation of CA by agents in $\mathcal{A}rgue_{Id(Ag^a)}$. The argumentative process for CA of Ag^a finishes when the acceptability of CA with respect to arguments from $\mathcal{A}rgue_{Id(Ag^a)}$ is established.

Independently of which situation occurs for each $Ag^a \in \mathcal{A}rgue_{Id(Ag)}$, if there exists at least one acceptable argument CA from $\mathcal{A}rgue_{Id(Ag)}$ against the available argument A of Ag, then A is not acceptable (with respect to $\mathcal{A}rgue_{Id(Ag)}$); otherwise, A is an acceptable argument. The following example illustrates the above informal description.

Example 30 *Assume the multi-agent setting*

$$\mathcal{A} = \{Ag_1, Ag_2, Ag_3, Ag_4, Ag_5, Ag_6\}$$

and the figure below which represents the cooperative and argumentative relations between such agents. For simplicity, we do not include in the graph the dashed loops showing that each agent argues and cooperates with itself.



To know about the acceptability of arguments for a in Ag_1: Ag_1 should have an argument for the objective literal a that must be acceptable w.r.t. $\mathcal{A}rgue_1 = \{1, 2, 3\}$; since Ag_1 has an argument $A_1^s(a) = (1, [c; a \leftarrow not\ \ b, c])$ we must check whether Ag_2 or Ag_3 have an argument against $A_1^s(a)$.

- *Ag_3 does not have any argument against $A_1^s(a)$ because it has only a partial argument for b and there is no argument from $\mathcal{C}ooperate_3$ to complete it.*
- *Ag_2 has a partial argument for b that can be completed by Ag_4's argument for f, i.e. $A_2^s(b) = (2, [f; b \leftarrow not\ \ d, f]) \in (2, [b \leftarrow not\ \ d, f]) + (4, [f])$. Still Ag_5 may have an argument against $A_2^s(b)$:*
  - *Ag_5 has the argument $A_5^s(d) = (5, [d \leftarrow not\ \ g])$, but now agent Ag_6 has an argument against $A_5^s(d)$.*
    - *Ag_6 has the argument $A_6^s(g) = (6, [g])$, which is acceptable because there is no argument from $\mathcal{A}rgue_6$ against it.*

    Thus, $A_5^s(d)$ is not acceptable because it is attacked by $A_6^s(g)$.

  Since Ag_5 has no acceptable argument against $A_2^s(b)$, $A_2^s(b)$ is an acceptable argument w.r.t. arguments from $\mathcal{A}rgue_2$.

Finally, $A_1^s(a)$ is not acceptable because there is at least one acceptable argument from $\mathcal{A}rgue_1$ against it, viz. $A_2^s(b)$.

We proceed by exposing the required definitions supporting this informal description. First of all, it is necessary to determine the available arguments that can be used to attack. As expected, only complete arguments in $\mathcal{A}rgs(\mathcal{A})$ should be considered. These arguments are called *attacking arguments*.

Definition 60 (Attacking Argument) *Let $\mathcal{A}$ be a MAS, α an agent's identity in $\mathcal{A}$, and $S \subseteq \mathcal{A}rgs(\mathcal{A})$. (α, Seq) is an* attacking argument *given S iff it is a complete argument and belongs to $Av(S)$. If (α, Seq) is either an s-argument or a w-argument, we refer to it by s-attacking or w-attacking argument, respectively.*

Example 31 *Consider the multi-agent setting $\mathcal{A}$ in Example 30. The least set of available arguments of $\mathcal{A}$ given $LA^s(\mathcal{A})$ is*

$$\begin{aligned} Av(LA^s(\mathcal{A})) = \{\ & (1, []),\ (1, [c]),\ (1, [a \leftarrow not\ \ b, c]),\ (1, [c; a \leftarrow not\ \ b, c]),\\ & (2, []),\ (2, [f]),\ (2, [b \leftarrow not\ \ d, f]),\ (2, [f; b \leftarrow not\ \ d, f]),\\ & (3, []),\ (3, [b \leftarrow d]),\ (4, []),\ (4, [f]),\ (5, []),\ (5, [d \leftarrow not\ \ g]),\\ & (6, []),\ (6, [g])\ \ \} \end{aligned}$$

and the set of attacking arguments given $LA^s(\mathcal{A})$ is

$$\begin{aligned} Atts = \{\ & (1, [c]),\ (1, [c; a \leftarrow not\ \ b, c]), (2, [f]),\ (2, [f; b \leftarrow not\ \ d, f]),\\ & (4, [f]),\ (5, [d \leftarrow not\ \ g]),\ (6, [g])\ \ \} \end{aligned}$$

Intuitively, both strong and weak arguments can be attacked in the same way. Since a (weak or strong) argument makes assumptions, other arguments for the complement of one such assumption may attack it. In other words, an argument with *not* L can be attacked by arguments for L. This definition of attack encompasses the case of arguments that are directly conflicting, e.g. an argument for L (with *not* $\neg L$) can be attacked by an argument for $\neg L$. Moreover, any weak argument $A^w_\alpha(L) = (\alpha, Seq^w_L)$ (and also a strong argument $A^s_\alpha(L) = (\alpha, Seq^s_L)$ which verifies *not* $\bot \in Assump(Seq^s_L)$) can be attacked by every argument for $\bot$. However, it does not make sense to attack arguments for objective literals if they do not lead to *falsity*. By "an objective literal L leads to *falsity*" we mean that there is an argument $A_\alpha(L)$ such that $A_\beta(\bot)$ is built based on such an argument, e.g.

$$A^s_\beta(\bot) : A^s_\alpha(L) + [\bot \leftarrow L, not\ \ L']$$

We only consider objective literals that are in the body of the rule for $\bot$ because these literals immediately lead to *falsity* and we say they are *directly conflicting with* $A_\beta(\bot)$. We assume that the involvement of other objective literals is not as strong as that of in the body of the denial[6].

We adapt the definition of *attack* from the centralized proposal (see Def. 41) by defining that an available argument of an agent Ag with an assumption *not* L can be attacked by an argument for L, only if that argument is from $\mathcal{A}rgue_{Id(Ag)}$. For the following definition, we use the centralized proposal's definitions of *Directly Conflicting via* $A_\bot$ and *Assumptions* (cf. Def. 40 and Def. 39, respectively). These sets are denoted by $DC(Seq_L)$ and $Assump(Seq_L)$, where Seq_L is a well-defined sequence for a literal L. We want to define attack in terms of both attacking and available arguments. However, we still need to determine which attacking arguments can be used to attack available ones. Care must be taken to prevent cycles between these definitions.

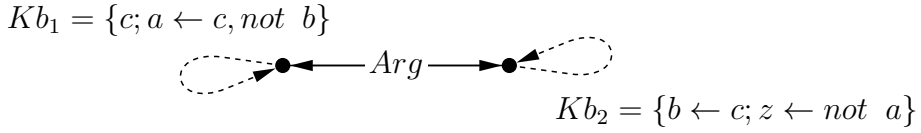
Definition 61 (Attack) *Let $\mathcal{A}$ be a MAS, α and β be agents identifiers in $\mathcal{A}$, and $\mathcal{A}rgue_\alpha$ the α's set of argumentative agents. An argument (β, Seq_{L_β}) [of β for L_β]* attacks *an argument (α, Seq_{L_α}) [of α for L_α] iff*

- $\beta \in \mathcal{A}rgue_\alpha$,
- *Seq_{L_β} is a well-defined sequence over R_β, or $Seq_{L_\beta} \in Seq_{L_\alpha} + Seq'_{L_\beta}$ where Seq'_{L_β} is a well-defined sequence for L_β over R_β, and*
- *L_β is the symbol $\bot$, not $\bot \in Assump(Seq_{L_\alpha})$ and $L_\alpha \in DC(Seq_{L_\beta})$, or L_β is an objective literal different from $\bot$ and not $L_\beta \in Assump(Seq_{L_\alpha})$.*

[6]We further assume they can be detected in a process of "belief revision", e.g. [DPS97, FKIS09]. However, a discussion of this issue is beyond the scope of this proposal.

Recall that, as with other argumentation based frameworks the semantics is defined based on a notion of acceptable arguments, where a set of arguments is acceptable if any argument attacking it is itself attacked by the set. Now, in this distributed setting, care must be taken about which arguments can be used to attack a set of arguments, and which arguments are available for being used to defend the attacks. Before presenting the definition of acceptable arguments we motivate for the definition in such a distributed setting. Moreover, note that the above definition of attack has a condition that foresees cases where "indirect cooperation" between argumentative agents is needed. The following example illustrates such a situation.

Example 32 *Consider the following graph representing a multi-agent setting $\mathcal{A}$ and assume that every argument in $LA(\mathcal{A})$ is a strong argument.*



The set of available arguments of $\mathcal{A}$ given $LA(\mathcal{A})$ is

$$\begin{aligned}\mathcal{A}rgs(\mathcal{A}) = \{\ \ & A_1^s(c) = [c],\\ & A_1^s(a) = [c; a \leftarrow c, not\ \ b],\\ & A_2^s(b) = [b \leftarrow c],\\ & A_2^s(z) = [z \leftarrow not\ \ a]\ \ \}\end{aligned}$$

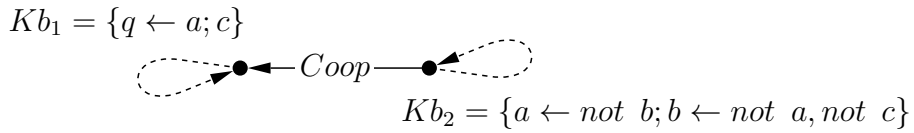
Moreover, from Definition 61, the complete argument $A_1^s(a)$ attacks $A_2^s(z)$ and the partial argument $A_2^s(b)$ attacks $A_1^s(a)$. However, we only want attacking arguments (i.e. complete arguments) to be used to determine the acceptability of an argument w.r.t. $\mathcal{A}rgs(\mathcal{A})$. Then, $A_2^s(b)$ will not be used and, consequently, $A_2^s(z)$ is not acceptable. Nevertheless, $A_1^s(a)$ has a rule for c that can be used to complete $A_2^s(b)$. If agent Ag_2 may 'use' such a rule from $A_1^s(a)$ to complete its partial argument $A_2^s(b)$, Ag_2 has an argument

$$(2, [c; b \leftarrow c])$$

that can be used against $A_1^s(a)$. Therefore, $A_2^s(z)$ is acceptable w.r.t. $\mathcal{A}rgs(\mathcal{A})$.

At this point, it is quite clear that we should evaluate available arguments of a multi-agent setting $\mathcal{A}$ to conclude which of them are acceptable with respect to a set S of arguments (that are already considered acceptable with respect to a set of arguments from $\mathcal{A}$). However, should an argument of an agent α be acceptable in

$\mathcal{A}rgue_\alpha$ if such an argument is to be used in a cooperation process? For instance, consider the following graph representing a multi-agent setting $\mathcal{A}$ and assume that both proposed and opposing arguments in $LA(\mathcal{A})$ are strong.



To have an acceptable argument for q, Ag_1 must complete its available argument for q, viz. $PA_1^s(q) = (1, [q \leftarrow a])$. Agent Ag_2 has an available argument for a, $A_2^s(a) = (2, [a \leftarrow not\ \ b])$. However, Ag_2 has also an attacking argument $A_2^s(b) = (2, [b \leftarrow not\ \ a, not\ \ c])$ against $A_2^s(a)$. Two possible approaches can deal with this situation:

1. since both arguments attack each other, neither $A_2^s(a)$ nor $A_2^s(b)$ are acceptable in $\mathcal{A}rgue_2$, and so $A_2^s(a)$ cannot be used to complete $PA_1^s(q)$; or

2. since there is no acceptable argument in $\mathcal{A}rgue_2$ attacking $A_1^s(a)$, it is defensible. Furthermore, $A_1^s(a)$ is used to complete $PA_1^s(q)$ and so the resulting available argument is

   $$A_1^s(q) = (1, [a \leftarrow not\ \ b; q \leftarrow a])$$

   However, $A_1^s(q)$ should be evaluated by $\mathcal{A}rgue_1$. Via cooperation, Ag_1 has an attacking argument

   $$A_1^s(b) = (1, [b \leftarrow not\ \ a, not\ \ c])$$

   against $A_1^s(q)$. But Ag_1 also has an attacking argument $A_1^s(c) = (1, [c])$ against $A_1^s(b)$ which no argument attacks. Thus, $A_1^s(c) = (1, [c])$ is acceptable and, consequently, $A_1^s(b)$ is not acceptable (both with respect to arguments from $\mathcal{A}rgue_1$). Therefore, $A_1^s(q)$ is acceptable with respect to arguments from $\mathcal{A}rgue_1$.

Intuitively, the second approach allows us to draw more acceptable arguments than the first one. Therefore, we follow the latter and define that for a given agent α in a multi-agent setting $\mathcal{A}$, an agent $\beta \in \mathcal{C}ooperate_\alpha$ cooperates with an available argument A under one of the following conditions: (i) A is not attacked by any argument from $\mathcal{A}rgue_\beta$, or (ii) A is attacked, but every attacking argument B against A is attacked by some argument from $\mathcal{A}rgue_\beta$. In both cases, A is considered a *defensible argument*. The following operator defines which are the defensible arguments, given a set of available arguments of a multi-agent setting.

Remark 47 *As in the centralized version, we use the notation p and o to distinguish the proposed argument from the opposing one, i.e. p (resp. o) is a (strong or weak) proposed (resp. opposing) argument.*

Definition 62 (Operator $Def_{p,o}(S)$) *Let $\mathcal{A}$ be a MAS, $\mathcal{A}rgs(\mathcal{A})$ be the set of available arguments of $\mathcal{A}$, and $S \subseteq \mathcal{A}rgs(\mathcal{A})$ be a set of p-arguments. $Def_{p,o}(S)$ is the set of all o-arguments of $\mathcal{A}rgs(\mathcal{A})$ that are not attacked by any attacking argument given S. Arguments in $Def_{p,o}(S)$ are called* defensible arguments.

Assume that arguments in the set of defensible arguments are opposing arguments, and every argument in a set of available arguments is a proposed argument. Now we can determine how available p-arguments are acceptable with respect to a set S of p-arguments from $\mathcal{A}rgs(\mathcal{A})$, such that S is a pre-defined set of acceptable arguments. In order to determine the set of acceptable arguments with respect to S, the following steps must be performed:

1. obtain the opposing arguments via $Def = Def_{p,o}(S)$. This encompasses the following two sub-steps:
   (a) get the set $Atts$ of p-attacking arguments given S, i.e. the complete arguments in $Av(S)$. This sub-step also allows p-partial arguments in S to be completed by arguments in S and so new p-arguments are built;
   (b) reject o-arguments in $\mathcal{A}rgs(\mathcal{A})$ that are attacked by arguments in $Atts$.
2. obtain the proposed (partial or complete) arguments given S, i.e. $Av(S)$. This sub-step also allows p-partial arguments to be completed by arguments in S and so new p-arguments are built.
3. determine which are (i) the opposing arguments attacking some proposed argument and (ii) the opposing arguments attacked by arguments in S.

Definition 63 (Acceptable Argument) *Let $\mathcal{A}$ be a MAS, $\mathcal{A}rgs(\mathcal{A})$ be the set of arguments of $\mathcal{A}$, $S \subseteq \mathcal{A}rgs(\mathcal{A})$ be a set of p-arguments, and α, β, and γ be agent's identifiers in $\mathcal{A}$. A p-argument (α, Seq_L) for a literal L is acceptable$_{p,o}$* w.r.t. *S iff (i) it is either a local argument, or it belongs to the set of available arguments given S; and (ii) for any o-attacking argument $(\beta, Seq_{L'})$ for a literal L' given $Def_{p,o}(S)$: if $(\beta, Seq_{L'})$ attacks (α, Seq_L) then there exists a complete p-argument $(\gamma, Seq_{L''})$ for a literal L'' in S that attacks $(\beta, Seq_{L'})$.*

Similar to the centralized proposal, we formalize the concept of acceptable arguments with a fixpoint approach and also define a characteristic function $p\ o$ of multi-agent setting $\mathcal{A}$ over a set of acceptable arguments S as follows:

Definition 64 (Characteristic Function) *Let $\mathcal{A}$ be a MAS, $\mathcal{A}rgs(\mathcal{A})$ be the set of available arguments of $\mathcal{A}$, and $S \subseteq \mathcal{A}rgs(\mathcal{A})$ be a set of p-arguments. The* characteristic function *p o of $\mathcal{A}$ over S is:*

$$F_{\mathcal{A}}^{p,o} : 2^{\mathcal{A}rgs(\mathcal{A})} \rightarrow 2^{\mathcal{A}rgs(\mathcal{A})},$$
$$F_{\mathcal{A}}^{p,o}(S) = \{A \in \mathcal{A}rgs(\mathcal{A}) \mid A \text{ is } acceptable_{p,o} \text{ w.r.t. } S\}$$

We can see that, if an argument A is $acceptable_{p,o}$ w.r.t. S, A is also $acceptable_{p,o}$ w.r.t. any superset of S. In fact, it can be shown that $Def_{p,o}(S)$ is anti-monotonic, and so $F_{\mathcal{A}}^{p,o}$ is monotonic.

Lemma 48 *$Def_{p,o}(S)$ is anti-monotonic*

Proof. Let S^1 and S^2 be two sets of arguments of a multi-agent setting $\mathcal{A}$, such that $S^1 \subseteq S^2$, and let $Arg \in Def_{p,o}(S^2)$. By definition of $Def_{p,o}$, Arg is not attacked by any attacking argument given S^2, i.e. by any complete argument in $Av(S^2)$. Clearly, $Av(S^2) \supseteq Av(S^1)$, and so Arg is not attacked by any complete argument in $Av(S^1)$, i.e. $Arg \in Def_{p,o}(S^1)$ ■

Proposition 49 *$F_{\mathcal{A}}^{p,o}$ is monotonic*

Proof. Let Arg' be an arbitrary argument that attacks Arg and belongs to $Def_{p,o}(S^2)$, and $S' \subseteq S^2$. By Lemma 48, $Arg' \in Def_{p,o}(S^1)$ and so, since Arg is $acceptable_{p,o}$ w.r.t. S^1, there exists $Arg'' \in S^1$ attacking Arg'. Since, by hypothesis, $S^1 \subseteq S^2$, $Arg'' \in S^2$. Thus, every argument in $Def_{p,o}(S^2)$ attacking Arg is attacked by an argument in S^2, i.e. $Arg \in F_{\mathcal{A}}^{p,o}(S^2)$ ■

Being monotonic, it is guaranteed that $F_{\mathcal{A}}^{p,o}$ always has a least fixpoint (according to the set inclusion ordering over sets of arguments):

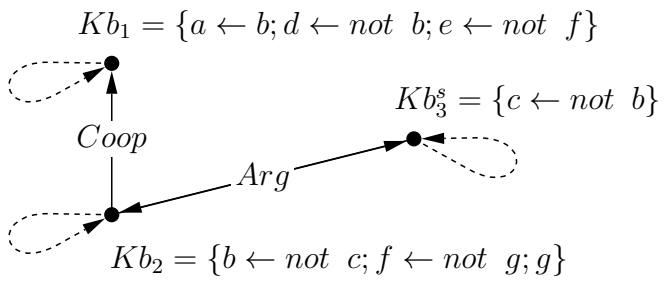
Proposition 50 *Define for any $\mathcal{A}$ the following sequence of sets of arguments:*

$$\begin{aligned} S^0 &= \emptyset \\ S^{i+1} &= F_{\mathcal{A}}^{p,o}(S^i) \end{aligned}$$

$F_{\mathcal{A}}^{p,o}$ is monotonic, and so there must exist a smallest λ such that S^λ is a fixpoint of $F_{\mathcal{A}}^{p,o}$, and $S^\lambda = lfp(F_{\mathcal{A}}^{p,o}) = F_{\mathcal{A}}^{p,o\uparrow\omega}$.

Proof. The result follows immediately from the monotonicity of $F_{\mathcal{A}}^{p,o}$, given the well-known Knaster-Tarski Theorem [Tar55] ■

Example 33 *Consider the following graph representing a multi-agent setting $\mathcal{A}$.*



In this example we show how to obtain $lfp(F_{\mathcal{A}}^{s,s}(\emptyset))$. *First of all, we determine the set of strong local arguments of* $\mathcal{A}$*:*

$$\begin{aligned} LA^s(\mathcal{A}) = \{ \quad & (1,[]),(1,[a \leftarrow b]),(1,[d \leftarrow not\ \ b]),(1,[e \leftarrow not\ \ f]), \\ & (2,[]),(2,[b \leftarrow not\ \ c]),(2,[f \leftarrow not\ \ g]),(2,[g]), \\ & (3,[]),(3,[c \leftarrow not\ \ b])\} \end{aligned}$$

and the set of available arguments of $\mathcal{A}$ *given* $LA^s(\mathcal{A})$*:*

$$\begin{aligned} \mathcal{A}rgs(\mathcal{A}) = \quad & LA^s(\mathcal{A}) \cup \\ & \{(1,[b \leftarrow not\ \ c]),(1,[b \leftarrow not\ \ c; a \leftarrow b]),(1,[f \leftarrow not\ \ g]),(1,[g])\} \end{aligned}$$

- *Let* $S^0 = \emptyset$. *Since* $Atts^0 = \emptyset$, *the set of opposing arguments is*

$$\begin{aligned} Def^0 = Def_{s,s}(S^0) = \{ \quad & (1,[d \leftarrow not\ \ b]),(1,[e \leftarrow not\ \ f]), \\ & (1,[b \leftarrow not\ \ c]),(1,[b \leftarrow not\ \ c; a \leftarrow b]), \\ & (1,[f \leftarrow not\ \ g]),(1,[g]),(2,[b \leftarrow not\ \ c]), \\ & (2,[f \leftarrow not\ \ g]),(2,[g]),(3,[c \leftarrow not\ \ b])\} \end{aligned}$$

  i.e. all complete arguments in $\mathcal{A}rgs(\mathcal{A})$. *The set of proposed arguments is* $LA^s(\mathcal{A})$, *resulting from* $Av(S^0)$. *Then we determine the following attacks*

opposing argument	*proposed argument*
$(1,[b \leftarrow not\ \ c])$	$(1,[d \leftarrow not\ \ b])$
$(1,[f \leftarrow not\ \ g])$	$(1,[e \leftarrow not\ \ f])$
$(3,[c \leftarrow not\ \ b])$	$(2,[b \leftarrow not\ \ c])$
$(2,[g])$	$(2,[f \leftarrow not\ \ g])$
	$(2,[g]$
$(2,[b \leftarrow not\ \ c])$	$(3,[c \leftarrow not\ \ b])$
	$(1,[a \leftarrow b])$
	$(1,[])$
	$(2,[])$
	$(3,[])$

  So $S^1 = F_{\mathcal{A}}^{s,s}(S^0) = \{(2,[g]),(1,[a \leftarrow b]),(1,[]),(2,[]),(3,[])\}$;

- *since* $Atts^1 = \{(2, [g])\}$, $(2, [f \leftarrow not\ \ g]) \in \mathcal{A}rgs(\mathcal{A})$ *is rejected and so*

$$Def^1 = Def_{s,s}(S^1) = \{ \begin{array}{l} (1, [d \leftarrow not\ \ b]), (1, [e \leftarrow not\ \ f]), \\ (1, [b \leftarrow not\ \ c]), (1, [b \leftarrow not\ \ c; a \leftarrow b]), \\ (1, [f \leftarrow not\ \ g]), (1, [g]), (2, [b \leftarrow not\ \ c]), \\ (2, [g]), (3, [c \leftarrow not\ \ b])\} \end{array}$$

  The set of proposed arguments is $Av(S^1) = S^1 \cup \{(1, [g])\}$. *There is no argument in* Def^1 *against* $(1, [g])$, *so* $S^2 = F^{s,s}_{\mathcal{A}}(S^1) = S^1 \cup \{(1, [g])\}$;

- *since* $Atts^2 = \{(2, [g]), (1, [g])\}$,

$$Def^2 = \mathcal{A}rgs(\mathcal{A}) - \{(2, [f \leftarrow not\ \ g]), (1, [f \leftarrow not\ \ g])\}$$

  The proposed arguments are obtained by $Av(S^2) = S^2$. *Despite the fact the opposing argument* $A^s_1(f) = (1, [f \leftarrow not\ \ g])$ *attacks the proposed argument* $(1, [e \leftarrow not\ \ f])$, $A^s_1(f)$ *is attacked by the acceptable argument* $(1, [g])$. *So* $S^3 = F^{s,s}_{\mathcal{A}}(S^2) = S^2 \cup \{(1, [e \leftarrow not\ \ f])\}$;

- *since* $F^{s,s}_{\mathcal{A}}(S^3) = S^3$, *we can say that the acceptable arguments of* $\mathcal{A}$ *are in*

$$lfp(F^{s,s}_{\mathcal{A}}(\emptyset)) = \{ \begin{array}{l} (2, [g]), (1, [a \leftarrow b]), (1, []), \\ (2, []), (3, []), (1, [g]), (1, [e \leftarrow not\ \ f])\} \end{array}$$

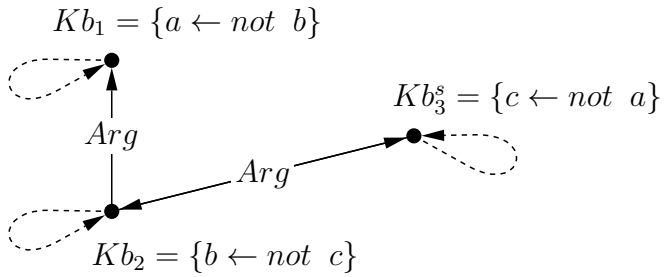
By knowing the set S of all acceptable arguments of $\mathcal{A}$, we can split all complete arguments from $\mathcal{A}rgs(\mathcal{A})$ into three classes: justified arguments, overruled arguments or defensible arguments. An argument A is *justified* when A is in S. An argument A is *overruled* when A is attacked by at least one argument in S. Finally, an argument A is *defensible* when A is attacked by an argument $B \in \mathcal{A}rgs(\mathcal{A})$, and neither A nor B are attacked by acceptable arguments.

Definition 65 (Justified, Overruled, or Defensible Argument) *Let* $\mathcal{A}$ *be a MAS,* $\mathcal{A}rgs(\mathcal{A})$ *be the set of available arguments of* $\mathcal{A}$, *and* $F^{p,o}_{\mathcal{A}}$ *be the characteristic function p o of* $\mathcal{A}$. *A complete p-argument for a literal L of an agent with identity* α *is:*

- justified$^{p,o}_{\mathcal{A}}$ *iff it is in* $lfp(F^{p,o}_{\mathcal{A}})$;
- overruled$^{p,o}_{\mathcal{A}}$ *iff there exists a justified*$^{o,p}_{\mathcal{A}}$ *o-argument for a literal* L' *of an agent* β *in* $\mathcal{A}$ *attacking it;*
- defensible$^{p,o}_{\mathcal{A}}$ *iff it is neither justified*$^{p,o}_{\mathcal{A}}$ *nor overruled*$^{p,o}_{\mathcal{A}}$.

We denote the $lfp(F^{p,o}_{\mathcal{A}})$ *by* $JustArgs^{p,o}_{\mathcal{A}}$.

Example 34 *Consider the following graph representing a multi-agent setting $\mathcal{A}$.*



In this example we show how to obtain $lfp(F_{\mathcal{A}}^{s,s})$. First of all, we determine the set of strong local arguments of $\mathcal{A}$:

$$LA^s(\mathcal{A}) \;=\; \left\{ \begin{array}{ll} (1,[]), & (1,[a \leftarrow not\ \ b]), \\ (2,[]), & (2,[b \leftarrow not\ \ c]), \\ (3,[]), & (3,[c \leftarrow not\ \ a]) \end{array} \right\}$$

and the set of available arguments of $\mathcal{A}$ given $LA^s(\mathcal{A})$, i.e $\mathcal{A}rgs(\mathcal{A}) = LA^s(\mathcal{A})$

- *let $S^0 = \emptyset$. Since $Atts^0 = \emptyset$, the set of opposing arguments is*

  $$Def^0 = Def_{s,s}(S^0) = \{(1,[a \leftarrow not\ \ b]), (2,[b \leftarrow not\ \ c]), (3,[c \leftarrow not\ \ a])\}$$

  The set of proposed arguments is $LA^s(\mathcal{A})$, resulting from $Av(S^0)$. Then we determine the following attacks

opposing argument	*proposed argument*
$(2,[b \leftarrow not\ \ c])$	$(1,[a \leftarrow not\ \ b])$
$(3,[c \leftarrow not\ \ a])$	$(2,[b \leftarrow not\ \ c])$
	$(3,[c \leftarrow not\ \ a])$
	$(1,[]), (2,[]), (3,[])$

  So $S^1 = F_{\mathcal{A}}^{s,s}(S^0) = \{(3,[c \leftarrow not\ \ a]), (1,[]), (2,[]), (3,[])\}$;

- *since $Atts^1 = \{(3,[c \leftarrow not\ \ a])\}$,*

  $$Def^1 = Def_{s,s}(S^1) = \{(1,[a \leftarrow not\ \ b]), (3,[c \leftarrow not\ \ a])\}$$

  The set of proposed arguments is $Av(S^1) = S^1$. Despite the fact that the opposing argument $A_2^s(b) = (2,[b \leftarrow not\ \ c])$ attacks the proposed argument $(1,[a \leftarrow not\ b])$, $A_2^s(b)$ is attacked by the acceptable argument $(3,[c \leftarrow not\ a])$, so

  $$S^2 = F_{\mathcal{A}}^{s,s}(S^1) = S^1 \cup \{(1,[a \leftarrow not\ \ b])\}$$

- *since* $F_{\mathcal{A}}^{s,s}(S^2) = S^2$, *the set of justified*$_{\mathcal{A}}^{s,s}$ *arguments is*

$$JustArgs_{\mathcal{A}}^{s,s} = \{(3, [c \leftarrow not\ \ a]), (1, [a \leftarrow not\ \ b]), (1, []), (2, []), (3, [])\}.$$

Argument $(2, [b \leftarrow not\ \ c])$ *is overruled*$_{\mathcal{A}}^{s,s}$ *because it is attacked by the justified*$_{\mathcal{A}}^{s,s}$ *argument* $(3, [c \leftarrow not\ \ a])$. *No argument in* $\mathcal{A}rgs(\mathcal{A})$ *is defensible*$_{\mathcal{A}}^{s,s}$.

4.4 Properties

Here we assume very little about the sets of argumentative and cooperative agents of an agent. By imposing some restriction on these sets different properties of the whole setting can be obtained. In particular, as expected, if all agents in a multi-agent setting $\mathcal{A}$ argue and cooperate with all others, then the result is exactly the same as having a single agent with the whole knowledge, i.e. we obtain the same results of the centralized proposal:

Theorem 51 *Let* $\mathcal{A}$ *be a MAS over* $\mathcal{L}$ *and* Ids *such that for every agent* $Ag \in \mathcal{A}$: $\mathcal{C}ooperate_{Id(Ag)} = Ids$ *and* $\mathcal{A}rgue_{Id(Ag)} = Ids$, *and* $F_{\mathcal{A}}^{p,o}$ *be the characteristic function p o of* $\mathcal{A}$. *Let* $P = \{< \beta, Kb_\beta, \{\beta\}, \{\beta\} >\}$ *such that*

$$Kb_\beta = \bigcup_{\alpha_i\ \in\ Ids} Kb_{\alpha_i}$$

and $F_P^{p,o}$ *be the characteristic function p o of* P.

Then, for every agent $\alpha_i \in Ids$: $(\alpha_i, Seq) \in lfp(F_{\mathcal{A}}^{p,o})$ *iff* $(\beta, Seq) \in lfp(F_P^{p,o})$.

Proof. The proof of this theorem follows easily from the very form of the definitions. In fact, it is easy to see that all the definitions in this Chapter collapse into the corresponding ones in the previous Chapter in case the agents only agree and cooperate with themself ■

Corollary 52 *If* $\mathcal{A}$ *is as in Theorem 51 then for any pair of agents in* $\mathcal{A}$, *with identifiers* α_i *and* α_j :, $(\alpha_i, Seq) \in lfp(F_{\mathcal{A}}^{p,o})$ *iff* $(\alpha_j, Seq) \in lfp(F_{\mathcal{A}}^{p,o})$.

However, if not all agents cooperate and argue with other, the semantics at one agent can be different from that of the union, as desired:

Example 35 *Consider* $\mathcal{A} = \{Ag_1, Ag_2, Ag_3\}$ *such that each agent is*

$$\begin{array}{lll} Ag_1 & = < & 1, \{\ a \leftarrow not\ \ b\ \},\ \{1,2,3\},\ \{1,2,3\}\}\ > \\ Ag_2 & = < & 2, \{\ b \leftarrow not\ \ c\},\ \{1,2,3\},\ \{1,2,3\}\ > \\ Ag_3 & = < & 3, \{\ c \leftarrow not\ \ a\ \},\ \{1,2,3\},\ \{1,2,3\}\}\ > \end{array}$$

$$Def_{s,s}(\emptyset) = \{ \quad (1,[]),(2,[]),(3,[]),(1,[a \leftarrow not\ \ b]),$$
$$(2,[a \leftarrow not\ \ b]),(3,[a \leftarrow not\ \ b]),(1,[b \leftarrow not\ \ c]),(2,[b \leftarrow not\ \ c]),$$
$$(3,[b \leftarrow not\ \ c]),(1,[c \leftarrow not\ \ a]),(2,[c \leftarrow not\ \ a]),(3,[c \leftarrow not\ \ a])\}$$

The arguments $A^s(a)$, $A^s(b)$, and $A^s(c)$) are attacked[7]*. As there is no "counter-attack" to any of those attacks, $lfp(F_{\mathcal{A}}^{s,s}(\emptyset)) = JustArgs_{\mathcal{A}}^{s,s} = \emptyset$. No argument in $\mathcal{A}rgs(\mathcal{A})$ is overruled$_{\mathcal{A}}^{s,s}$, and all of arguments are concluded to be defensible$_{\mathcal{A}}^{s,s}$. However, we obtain a different result if we consider that*

$$\begin{array}{lll} Ag_1 & = < & 1, \{\ a \leftarrow not\ \ b\ \},\ \{1,2\},\ \{1,2\}\} > \\ Ag_2 & = < & 2, \{\ b \leftarrow not\ \ c\},\ \{2,3\},\ \{2,3\} > \\ Ag_3 & = < & 3, \{\ c \leftarrow not\ \ a\ \},\ \{3\},\ \{3\}\} > \end{array}$$

$$Def_{s,s}(\emptyset) = \{ \quad (1,[]),(2,[]),(3,[]),(1,[a \leftarrow not\ \ b]),(1,[b \leftarrow not\ \ c]),$$
$$(1,[c \leftarrow not\ \ a]),(2,[b \leftarrow not\ \ c]),(2,[c \leftarrow not\ \ a]),(3,[c \leftarrow not\ \ a])\}$$

and $lfp(F_{\mathcal{A}}^{s,s}(\emptyset)) = JustArgs_{\mathcal{A}}^{s,s} = \{A_2^s(c), A_3^s(c)\}$. Note here how the result differs also from agent to agent.

In the centralized version, a justified p-argument might also be *related to a contradiction* (cf. Def. 48). From such a definition, an argument can be contradictory, based on contradiction, or non-contradictory with respect to the set of justified arguments of an extended logic program with denials P, i.e. with $JustArgs_P^{p,o} = lfp(F_P^{p,o})$. However, both $JustArgs_P^{w,w}$ and $JustArgs_P^{w,s}$ are both conflict-free[8] and non-contradictory[9] (see proof of propositions 25, 26 and 27). Thus, every argument in both $JustArgs_P^{w,w}$ and $JustArgs_P^{w,s}$ is non-contradictory, i.e. it is not related to a contradiction at all. Furthermore, $F_P^{w,w}$ has more defensible arguments than $F_P^{w,s}$ (cf. proof of Prop. 28). Therefore, we obtain a consistent way of reasoning in a multi-agent setting $\mathcal{A}$ if we apply $F_{\mathcal{A}}^{w,w}$ over $\mathcal{A}rgs(\mathcal{A})$.

In contrast, $JustArgs_{\mathcal{A}}^{s,s}$ and $JustArgs_{\mathcal{A}}^{s,w}$ may be contradictory, i.e. they might have arguments related to a contradiction. However, to evaluate the acceptability of available arguments without considering the presence of *falsity* or both arguments for L and $\neg L$, the proposed arguments should be strong ones, and every opposing argument is a weak argument. Since $F_{\mathcal{A}}^{s,w}$ respects the 'Coherence Principle' of [PA92, ADP95], i.e. given that every opposing argument is a weak one,

[7]For simplicity, since every agent argues with every other, we omit agent identity of the arguments.

[8]A set S of arguments is conflict-free if there is no argument in S attacking an argument in S (cf. Def. 45).

[9]A set S of arguments is non-contradictory if neither an argument for *falsity* nor both arguments for L and $\neg L$ are in S (cf. Def. 46).

it can be attacked by any proposed argument for its explicit negation. Therefore, we obtain a paraconsistent way of reasoning in a multi-agent setting $\mathcal{A}$ if we apply $F_{\mathcal{A}}^{s,w}$ over $\mathcal{A}rgs(\mathcal{A})$. Moreover, a justified$_{\mathcal{A}}^{s,w}$ argument of an agent in $\mathcal{A}$ is related to a contradiction with respect to $JustArgs_{\mathcal{A}}^{s,w}$:

Definition 66 (Relation to a Contradiction) *Let $\mathcal{A}$ be a MAS, α and β be agents' identity in MAS, $\beta \in \mathcal{A}rgue_{\alpha}$, and $JustArgs_{\mathcal{A}}^{s,w}$ be the $lfp(F_{\mathcal{A}}^{s,w})$. A justified$_{\mathcal{A}}^{s,w}$ s-argument $A_{\alpha}^{s}(L) = (\alpha, Seq_L)$ is:*

- *contradictory$_{\mathcal{A}}^{s,w}$ if L is the symbol $\bot$, or there exists a justified$_{\mathcal{A}}^{s,w}$ s-argument $(\beta, Seq_{\bot})$ such that $L \in DC(Seq_{\bot})$, or there exists a justified$_{\mathcal{A}}^{s,w}$ s-argument $(\beta, Seq_{\neg L})$; or*
- *based-on-contradiction$_{\mathcal{A}}^{s,w}$ if $A_{\alpha}^{s}(L)$ is justified$_{\mathcal{A}}^{s,w}$, it does not exists a justified$_{\mathcal{A}}^{s,w}$ s-argument $(\beta, Seq_{\neg L})$ and $A_{\alpha}^{s}(L)$ is also overruled$_{\mathcal{A}}^{s,w}$; or*
- *non-contradictory$_{\mathcal{A}}^{s,w}$, otherwise.*

As already said, any agent's belief should be concluded only with respect to both sets of argumentative and cooperative agents with such an agent. Intuitively, we can conclude that different truth values of a given literal L over a multi-agent setting $\mathcal{A}$ might be obtained. It happens because it depends on which agent the literal L is inferred from, and also on what the specification of both sets of cooperative and argumentative agents is, given the overall agents in $\mathcal{A}$. Then, a truth value of an agent's conclusion in a (consistent or paraconsistent) way of reasoning is as follows:

Definition 67 (Truth Value of an Agent's Conclusion) *Let $\mathcal{A}$ be a MAS, α is an agent's identity of $\mathcal{A}$, $k \in \{s, w\}$, and L be an objective literal or the symbol $\bot$. L over $\mathcal{A}$ is:*

- false$_{\alpha}^{k,w}$ *iff for all argument of α for L: it is overruled$_{\mathcal{A}}^{k,w}$*
- true$_{\alpha}^{k,w}$ *iff there exists a justified$_{\mathcal{A}}^{k,w}$ argument of α for L. Moreover, L is*
  - contradictory$_{\alpha}^{k,w}$ *if L is the symbol $\bot$ or there exists a justified$_{\mathcal{A}}^{k,w}$ argument of α for $\neg L$*
  - based-on-contradiction$_{\alpha}^{k,w}$ *if it is both true$_{\alpha}^{k,w}$ and false$_{\alpha}^{k,w}$*
  - non-contradictory$_{\alpha}^{k,w}$, *otherwise.*
- undefined$_{\alpha}^{k,w}$ *iff L is neither true$_{\alpha}^{k,w}$ nor false$_{\alpha}^{k,w}$.*

Note at this point that truth is defined parametric of the agent. So, it is only natural that the truth value of a proposition may differ from agent to agent.

Proposition 53 *Let $k \in \{s, w\}$. L is undefined$^{k,w}_{\alpha}$ iff there is no justified$^{k,w}_{\mathcal{A}}$ argument of α for L and at least one argument of α for L is not overruled$^{k,w}_{\mathcal{A}}$.*

This paraconsistent semantics for multiple logic programs is in accordance with the paraconsistent well-founded semantics $WFSX_p$ [ADP95]. In fact, both coincide if there is a single program (or a set, in case all cooperate and argue with all other, cf. Theorem 51):

Theorem 54 ($WFSX_p$ semantics vs $F^{s,w}_{\mathcal{A}}$) *Let P be an ELP such that $\bot \notin \mathcal{H}(P)$, and let L be an objective literal in $\mathcal{H}(P)$. $L \in WFSX_p(P)$ iff L is true$^{s,w}_{\mathcal{A}}$, not $L \in WFSX_p(P)$ iff L is false$^{s,w}_{\mathcal{A}}$, and $\{L, not\ L\} \cap WFSX_p(P) = \emptyset$ iff L is undefined$^{s,w}_{\mathcal{A}}$.*

Proof. Follows directly from Theorem 51 and Theorem 33 above ■

Moreover, there is a relation between the consistent reasoning obtained with $F^{w,w}_{\mathcal{A}}$ and [Dun95]'s grounded (skeptical) extension:

Theorem 55 (Grounded extension vs $F^{w,w}_{\mathcal{A}}$) *Let P be an ELP such that $\bot \notin \mathcal{H}(P)$, L be an objective literal in $\mathcal{H}(P)$, B be the Ground Extension's characteristic function of P, (A_L, L) be an argument for L, and α be an agent's identity of $\mathcal{A}$.*

An argument $(A_L, L) \in lfp(B)$ iff $\exists(\alpha, Seq^w_L) \in lfp(F^{w,w}_{\mathcal{A}})$.

An argument $(\{not\ L\}, L) \in lfp(B)$ iff $\neg\exists(\alpha, Seq^w_L) \in gfp(F^{w,w}_{\mathcal{A}})$.

Proof. Follows directly from Theorem 51 and Theorem 36 above ■

4.5 Other Illustrative Examples

As put forth, the ability to associate argumentative and cooperative sets to each agent provides a flexible framework which, besides reflecting the possibly existing physical network, may serve for other purposes:

- For modelling knowledge over a hierarchy where each node of the hierarchy is represented by a Kb that cooperates with all its inferiors, and must argue with all its superiors.

- For modelling knowledge that evolves. Here the "present" can use knowledge from the "past" unless this knowledge from the past is in conflict with later knowledge. This can be modelled by allowing any present node to cooperate with its past nodes, and forcing any past node to argue with future nodes.

In these cases, it is important that the knowledge is not flattened, as in the union of all knowledge bases, and that the semantics is parametric on the specific Kb. I.e. it might happen that an argument is acceptable in a given (agent$_i$) Kb_i, and not acceptable in another (agent$_j$) Kb_j of the same system. This section shows that our proposal allows the modelling of a multi-agent setting with those different kinds of representation.

4.5.1 Representing Hierarchy of Knowledge

This example is derived from both the ADEPT project [JFJ+96] which developed negotiating agents for business process management and the argumentation framework [Sch99].

Example 36 (Business Process Management) *One process deals with the provision of customer quotes for networks adapted to the customer's needs. Four agents are involved in this process: (i) the* customer service division *(CSD), which makes the initial contact with the customer and delivers the quote eventually, (ii) the* vet customer *(VC) agent, which determines whether the customer is credit-worthy, (iii) the* design department *(DD), which does the design and costing of the requested network if it is not a portfolio item, and (iv) the* surveyor department *(SD), which may have to survey the customer site for the design department.*

Initially, a customer issues a request. The CSD gathers some data for this request, such as requirements, the equipment already installed at the customer site, and how important that client is. Before any action is taken, the CSD asks the VC to vet the customer. If the customer is not found credit-worthy, the process terminates and no quote is issued to the customer. If it is credit-worthy, the CSD checks whether the required network is a portfolio item with a previous quote. If positive, this quote is sent to the customer; otherwise, the design department is contacted. The DD develops its design and costing based on the information of given equipment held by the CSD. In many cases, this information may be out of date or not available at all, so the site has to be surveyed. In this case, the DD contacts the surveyors to do a survey. After the survey is done, the DD can design and cost the network. Then CSD can finally provide the customer quote.

CSD must not quote if the customer is not credit-worthy, which should be assumed by default. So, CSD should obtain an agreement with VC, which means that VC may counter-argue and give evidence for the credit-worthiness of the customer. In case credit is approved, if CSD does not have a portfolio item for the solicited quote, it needs a quote for it from DD. DD might do this task if SD does not argue that such a task is not important. If DD can do its task, it needs information held by CSD. Figure 4.4 illustrates the arguing and cooperating relation between such agents.

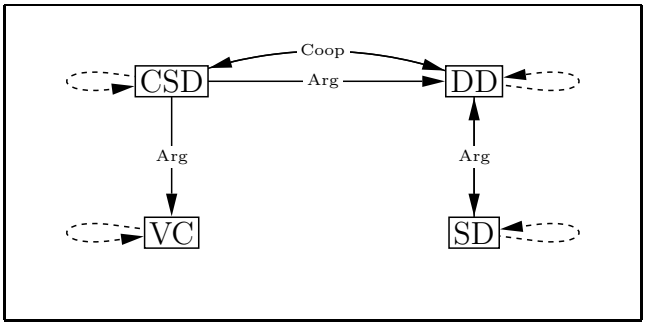


Figure 4.4: "Business Process Management"

In the following we present the corresponding agents' knowledge base as extended logic program with denials. Considering first the *Customer Service Division*, it knows about the client's equipment (called eq) and its requirements (called req). However, the description is not explicit with reference to them, so we assume that CSD has requirements 2 and 3, and equipment 2 and 3. Furthermore, CSD knows the customer is important. These can be represented as facts:

$$\begin{array}{ll} req2. & req3. \\ eq2. & eq3. \\ important. & \end{array}$$

Besides these facts about a particular client, CSD has general rules such as requirements 1, 2 and 3 together making up a portfolio and being quotable if a previous quote exists (otherwise, the DD has to prepare a quote).

$$portofolio \leftarrow req1, req2, req3.$$
$$quote \leftarrow portfolio, previousQuote.$$

CSD does not provide a quote if the client is not credit-worthy, which is assumed by default:

$$\neg quote \leftarrow not\ creditWorthy.$$

The *Vet Customer* knows the client is credit-worthy, i.e it has a fact

$$creditWorthy.$$

The *Design Department* knows that there is no need to survey the client site if the client has equipments 1, 2 and 3. It can be represented by the rule

$$\neg need2survey \leftarrow eq1, eq2, eq3.$$

In general, DD assumes that SD does a survey unless it is busy, which can be represented by the rule

$$survey \leftarrow not\ busySD$$

The quote of DD can be obtained by a simple design cost if there was no need to survey; otherwise, it is obtained by a complex design cost:

$$quote \leftarrow \neg need2survey, simpleDesignCost.$$
$$quote \leftarrow survey, complexDesigCost.$$
$$simpleDesignCost.$$
$$complexDesignCost.$$

Finally, the knowledge of *Surveyor Department* is fairly simple, i.e. its domain is its own business, and since it is lazy it derives that it is busy unless the customer is important. The following rule represents such a belief:

$$busySD \leftarrow not\ \ important.$$

The multi-agent setting presented in Example 36 is defined as follows

$$\mathcal{B}PM = \{CSD, DD, VC, SD\}$$

and the agents of $\mathcal{B}PM$ are represented as follows, with obvious abbreviations,

$$\begin{array}{ll} CSD = & < csd, Kb_{csd}, \{csd, vc, dd\}, \{csd, dd\} > \\ VC = & < vc, \{crWo\}, \{vc\}, \{vc\} > \\ DD = & < dd, Kb_{dd}, \{dd, sd\}, \{dd, csd\} > \\ SD = & < sd, \{bSD \leftarrow not\ \ imp\}, \{sd, dd\}, \{sd\} > \end{array}$$

where the knowledge bases of CSD and DD are

$$\begin{array}{ll} Kb_{csd} = \{ & re(2);\ re(3);\ eq(3);\ eq(2);\ imp;\ por \leftarrow re(1), re(2), re(3); \\ & \neg quo \leftarrow not\ \ crWo;\ quo \leftarrow por, preQuo\ \ \} \end{array}$$

$$\begin{array}{ll} Kb_{dd} = \{ & \neg n2s \leftarrow eq(1), eq(2), eq(3);\ sur \leftarrow not\ \ bSD;\ sDC;\ cDC; \\ & quo \leftarrow \neg n2s, sDC;\ quo \leftarrow sur, cDC\ \ \} \end{array}$$

Since such a system must have consistent conclusions we illustrate the results in a consistent reasoning, i.e. with weak proposing and opposing arguments. The truth value of the main conclusions are as follows: *important* is $true^{w,w}_{csd}$, *complexDesignCost* and *survey* are $true^{w,w}_{dd}$, $\neg creditWorthy$ is $true^{w,w}_{vc}$; $busySD$ is $false^{w,w}_{sd}$; both *quote* and $\neg quote$ are $undefined^{w,w}_{csd}$.

Finally, [Sch99]'s argumentation framework determines that an agent is defined by (i) a set of arguments, (ii) a set of predicate names defining the agent's domain of expertise, (iii) a flag indication whether the agent is credulous or skeptical, (iv) a set of cooperation partners, and (v) a set of argumentation partners. The above example illustrate a multi-agent setting similar to [Sch99]'s proposal, which means that our agents have (i), (iv) and (v).

4.5.2 Obtaining Conclusions at Different Periods of Time

We use an example model elaborated by [SPR98] based on the following contradiction found in the third act of Shakespeare's Hamlet:

> "Should Hamlet kill Claudius? Hamlet is unsure whether to kill Claudius — the assassin of Hamlet's father — or not. He argues that if he does kill him, Claudius, who is praying at that very moment, goes to heaven, and if he does not kill him, Hamlet's father is not revenged. A contradiction."

The model was developed in an argumentation framework, Ultima Ratio[10], which aims to formalize and to visualize agents' argumentation. An agent is composed of a set of arguments and assumptions. Facing a particular world, the agent's beliefs may be inconsistent, triggering a rational monologue to deal with the situation. Formally, [SPR98] defines a framework for argumentation based on extended logic programming under well-founded semantics. Given the specification, Ultima Ratio unfolds a process of argumentation in which arguments and counter-arguments are exchanged to detect conflicts and remove them. Ultima Ratio uses extended logic programming with denials, such that the denials characterize contradictory situations. There is also a definition of revisable assumptions in order to remove inconsistencies. Furthermore, Ultima Ratio is based on [PS97]'s proposal, and this system also distinguishes two kinds of attacks: on the conclusion of an argument (rebut), or on the premises of an argument (undercut). The system serves as decision support, and it is capable of detecting and removing contradictions and of deriving conclusions from the agent's arguments.

Our aim in this section is to show that we can also detect the conflicts and derive conclusions from the agent's set of arguments. This example illustrates how we obtain conclusions from Hamlet's knowledge in a particular point in, i.e. the conclusions of his beliefs in the past, in the present, and further in a possible future. How can we do it? By building a multi-agent setting, dubbed $\mathcal{H}amlet$, where each agent represents a period of Hamlet's lifetime and the respective 'knowledge'. Figure 4.5 illustrates the dependencies between those periods. To reach a conclusion about something in *past*, it is necessary to 'confirm' the truth value of such a conclusion in *present*, i.e. *past* should argue with *present*. However, *present* might need some information from *past* to complete its incomplete knowledge, i.e. *present* needs cooperation with *past*. The process of argumentation and cooperation between *present* and *future* is the same as described above.

The following example is our proposal for modelling Hamlet's conflict:

[10] See *http://www.sabonjo.de/texts.html*.

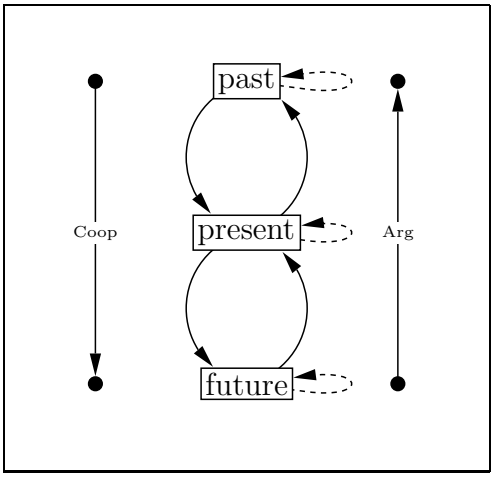


Figure 4.5: Hamlet's knowledge in periods of time

Example 37 (Hamlet Conflicts) *Hamlet realizes that Claudius is praying. This is represented by the fact*

$$praying(claudius).$$

Hamlet has a belief that Claudius would go to heaven if he kills Claudius while Claudius is praying:

$$inHeaven(X) \leftarrow kills(Y, X),\ praying(X).$$

Hamlet states that killing Claudius satisfies his desire for revenge:

$$takesRevengeOn(X, Y) \leftarrow goalRevenge(X, Y),\ kills(X, Y).$$

Hamlets starts another line of reasoning by mentioning the fact that Claudius killed Hamlet's father, when was the king:

$$killed(claudius, king).$$

Hamlet finds that he is not avenged if he sends Claudius to heaven:

$$\neg takesRevengeOn(X, Y) \leftarrow inHeaven(Y).$$

Besides this direct translation, further facts and rules are added which are mentioned throughout preceding scenes or which are given implicitly. The rule

$$goalRevenge(X, Y) \leftarrow closeRelationship(X, Z), killed(Y, Z), not\ reason(killed(Y, Z)).$$

expresses that someone wants revenge: person X wants to take revenge on person Y if Y killed person Z who was close to X, and there was no reason for Y to kill

Z. Left implicitly in the play is the fact that Hamlet and his father had a close relationship:

$$closeRelationship(myself, king).$$

In this scene, it is stated formally that Hamlet has a denial that he wants to take revenge and he does not take it

$$\bot \leftarrow goalRevenge(X, Y),\ not\ \ kills(X, Y).$$

Finally, Hamlet wants to kill Claudius, i.e. to assume the fact

$$kills(myself, claudius).$$

The decision will determine which kind of conflict Hamlet will have: a contradiction between $takesRevengeOn(myself, claudious)$ *and* $\neg takesRevengeOn(myself, claudious)$*, or the presence of* falsity*.*

The multi-agent setting presented in Example 37 is defined as

$$\mathcal{H}amlet = \{past, present, future\}$$

and each agent of $\mathcal{H}amlet$ should have some of the rules above. We consider most of the general rules as Hamlet's knowledge acquired before his father died, i.e. in the *past*. In the *present*, two events happen in a short period of time: Claudius kills Hamlet's father and Claudius is praying. After these events, Hamlet starts to be in conflict: "Does he take revenge on Claudius?" or "Does he not take it?" We consider such suppositions as events that might occur in the *future*. Nevertheless, the future might have the fact $kills(myself, claudius)$. Such a 'decision' will derive different conclusions from $\mathcal{H}amlet$. So, we illustrate Example 37 with two versions for the *future*'s knowledge base, viz. Kb^{k}_{fu} and Kb^{nk}_{fu}. The former version includes the fact $ki(m, c)$ and the latter does not.

$$Kb_{pa} = \{\ \ gR(X, Y) \leftarrow cR(X, Z),\ ked(Y, Z), not\ \ re(ked(Y, Z)); cR(m, k);\\ \bot \leftarrow gR(X, Y),\ not\ \ ki(X, Y);\ iH(X) \leftarrow ki(Y, X),\ pra(X)\ \ \}$$

$$Kb_{pr} = \{\ \ ked(c, k); pra(c)\ \ \}$$

$$Kb^{k}_{fu} = \{\ \ tRO(X, Y) \leftarrow gR(X, Y),\ ki(X, Y); \neg tRO(X, Y) \leftarrow iH(Y);\ ki(m, c)\ \ \}$$

$$Kb^{nk}_{fu} = \{\ \ tRO(X, Y) \leftarrow gR(X, Y),\ ki(X, Y); \neg tRO(X, Y) \leftarrow iH(Y)\ \ \}$$

By consequence, we have two versions for $\mathcal{H}amlet$:

$$\begin{array}{ll} \mathcal{H}amelt^{k} = \{ & < pa, Kb_{pa}, \{pa\}, \{pa, pr\}\}, \\ & < pr, Kb_{pr}, \{pr, pa\}, \{pr, fu\} >\}, \\ & < fu, Kb^{k}_{fu}, \{fu, pr\}, \{fu\} >\} \ \} \end{array}$$

$$\begin{array}{ll} \mathcal{H}amelt^{nk} = \{ & < pa, Kb_{pa}, \{pa\}, \{pa, pr\} >\}, \\ & < pr, Kb_{pr}, \{pr, pa\}, \{pr, fu\} >\}, \\ & < fu, Kb^{nk}_{fu}, \{fu, pr\}, \{fu\} >\} \ \} \end{array}$$

We illustrated here a multi-agent setting where each agent argues (resp. cooperates) with the next (resp. previous) agent. Nevertheless, it would possible to obtain a conclusion in a period of time without considering what happens after such a period. For instance, assume a multi-agent setting $\mathcal{A} = \{T_1, T_2, \ldots, T_n\}$ where each T_i $(1 \leq i \leq n)$ represents a different period of time in $\mathcal{A}$. In the case that we illustrate, if agent T_i argues with agent T_{i+1}, a truth value of conclusion L in the period T_i is obtained by considering what T_{i+1} knows about L (perhaps a counter-argument against an argument for L). On the other hand, if agent T_i does not argue with agent T_{i+1}, the truth value of L is obtained until period T_i without considering the fact that the arguments for L might be attacked by arguments in T_{i+1}.

Finally, in the next example, taken from [ALP+00], note how the cooperation is used to inherit rules from the past, and the argumentation to make sure that previous rules in conflict with later ones are overruled.

Example 38 *In this example we illustrate the usage of the proposed framework to reason about evolving knowledge bases. Each argumentative agent represents the knowledge (set of rules) added at a point of time. Moreover, each agent can cooperate with all agents representing past states, and has to argue with all agents representing future states. Consider agent Ag_1:*

$$\begin{array}{l} sleep \leftarrow \neg tv\_on. \\ tv\_on. \\ watch\_tv \leftarrow tv\_on. \end{array}$$

It is easy to see that with this knowledge in Ag_1, there is a justified$^{w,w}_{\mathcal{A}}$ argument for watch_tv and there is only a partial argument for sleep. The knowledge is then updated, in Ag_2, by adding the rules:

$$\begin{array}{l} \neg tv\_on \leftarrow power\_failure. \\ power\_failure. \end{array}$$

The reader can check that, for Ag_2 the previous argument for watch_tv is now overruled$^{w,w}_{\mathcal{A}}$, and that the argument for sleep is justified$^{w,w}_{\mathcal{A}}$. Now if another update comes, e.g stating $\neg power\_failure$, in Ag_3 the argument for sleep is again overruled$^{w,w}_{\mathcal{A}}$, and for watch_tv justified$^{w,w}_{\mathcal{A}}$, as expected.

4.6 On the implementation of the proposed semantics

Although the definition of a correct formal proof procedures for the declarative semantics just exposed, and its corresponding implementation, is outside the scope of this thesis, it is important at least to highlight how such a proof procedures may be defined, and how an implementation for the semantics can be obtained. In fact, we already have some preliminary results on the definition of the proof procedures, and have made some experiments on the implementation of a system for the semantics.

We start this section by providing some intuition on proof procedures for the semantics just exposed, and continue with a sketch of an algorithm realizing such a procedure. We then draw some consideration on the implementation that result from the experiments made.

It is important to make clear from the beginning that we are interest in a query-driven system in a multi-agent setting. The query should be any objective literal L, and the system should return its truth value over a multi-agent system $\mathcal{A}$ where each agent's knowledge base is represented as an extended logic program. Furthermore, when the literal L is true, the system should be able to provide the argument that supports L.

For this distributed semantics, we do not follow the idea of the operational semantics of the centralized proposal presented in Chapter 3, which defines a dialogue as a sequence of p and o moves of proposed arguments and opposing arguments, respectively. Instead, we must rely on a system which deals with negotiation processes.

A negotiation process for an objective literal L of an agent α involves both argumentative and cooperative processes. A cooperative process is started to complete a local partial argument with arguments from cooperative agents. An argumentative process is started to evaluate the acceptability of a complete argument with respect to arguments from argumentative agents. Both processes are interleaved, i.e. an argument from a cooperative agent has to be evaluated in an argumentation process, and a partial argument from an argumentative agent is completed by a cooperation process. For doing so,

- we have to deal with the fact that an agent has both cooperative and argu-

mentative "behaviors" in a negotiation process. Moreover, the agent is either a proponent or an opponent during the process. In other words, cooperative agents have to complete either proponent or opponent partial arguments, and argumentative agents have to evaluate both proponent and opponent arguments;

- the (argumentative and cooperative) agents have to exchange messages and so we have to define a communication protocol; and

- we have to define how to represent and control the negotiation process distributed through a set of agents.

For dealing with both argumentative and cooperative processes, we informally define two kinds of agent's dialogue, viz. argumentative dialogue and cooperative dialogue, as follows:

- during an argumentative dialogue the agents exchange *proposes*, *opposes*, and *agreements*. A cooperative dialogue involves *asks* and *replies*. A propose (resp. an ask) is sent to every argumentative (resp. cooperative) agent. However, the answer of such a message is only between the sender and a recipient, i.e. between the proponent (resp. asking) agent and one of its opponent (resp. cooperative) agent;

- an objective literal L is represented as a tree such that the root node is an empty argument for L, and left nodes are (partial or complete) argument for L resulting from cooperation process. The branches represent all possible ways to build such arguments.

- an *argumentative dialogue* for an objective literal L is seen as a set of both p and o trees for building proposed arguments and opposing arguments, respectively; a *cooperative dialogue* for an objective literal L is seen as a set of p (resp. o) trees for building proposed (resp. opposing) arguments.

Therefore, a cooperative process for a partial argument PA of an agent α is a set of cooperative dialogues started in every α's cooperative agent for every "unknown" L_i in PA; an argumentative process for a complete argument A of an agent α is a set of argumentative dialogues started in every α's argumentative agent for every default literal in A. Moreover, a multi-agent negotiation process is seen a forest of p and o trees distributed through the agents.

Given these general consideration, we can now start sketching an algorithm for the negotiation process of an agent α. First, we present some definitions/operators that we use in the algorithm:

- A (resp. PA) denotes a local complete argument (resp. local partial argument) (α, Seq_L) of agent α for L (cf. Def. 56 in Section 3.2)
- $plus((\alpha, Seq_L), Seq)$ returns a set of (complete and partial) arguments of α for L resulting from $Seq_L + Seq$ (cf. Def. 58 on Section 4.3);
- $player \in \{p, o\}$ such that p denotes a proponent player whereas o denotes an opponent player;
- $k \in \{s, w\}$ such that s denotes a strong argument whereas w denotes a weak argument;
- $Kp(player)$ returns the $player$'s kind of argument, i.e. weak or strong argument. For simplification, we suppress the kind of the argument by assuming that every argument of a $player$ agent is $Kp(player)$;
- $Op(player)$ returns the player's opposer, i.e. $Op(p)$ returns o whereas $Op(o)$ returns p;
- $Un(L, (\alpha, Seq))$ returns the set of "unknown conclusions" L_i of (α, Seq), i.e. every objective literal $L_i \in Conc(Seq)$ (cf. Def. 39 in Section 3.2) for which there is no rule $L_i \leftarrow Body$ in Seq. $\{L\} = Un(L, (\alpha, []))$;
- $Assump$ is a set of objective literals such that $not\ Assump = Assump(Seq_L)$. $Assump(Seq_L)$ returns the set of all default literals appearing in the bodies of rules in Seq_L (cf. Def. 39 in Section 3.2);
- $Atts((\alpha, Seq_L), player)$ returns the set of all possible "conclusions" against the argument for L. If $Kp(player) = s$ then $Atts(A, player) = Assump$ else $Atts = \{\neg L, falsity\} \cup Assump$;
- an argument CA attacks an argument A (cf. Def.61 in Section 3.2);
- A is $acceptable_{p,o}$ w.r.t. $\mathcal{A}rgue_\alpha$ (cf. Def. 63 in Section 3.2).

A negotiation process starts when an agent α receives an **ask(L)** for an objective literal L from an external agent[11]. If α has only a local partial argument PA for L then α starts a **cooperative process** for PA, as proponent, by sending the message $ask(\alpha, L_i, p)$ for every objective literal $L_i \in Un(L, PA)$ to every $\beta \in \mathcal{C}ooperate_\alpha$. When α has a complete local argument A for L – built by itself or via cooperative process for L – then α starts an **argumentative process** for A, as proponent, by sending the message $propose(\alpha, L, A, p)$ to every $\beta \in \mathcal{A}rgue_\alpha$;

[11] An external agent does not belong to the Multi-agent Setting.

otherwise, α sends to the external agent a $reply(L, none)$. If the argumentation process succeeds then α sends to the external agent a $reply(L, A)$; otherwise, it sends a $reply(L, none)$.

- **Cooperative process (for a local partial argument PA for L in α as $player$)**

  A cooperative process **succeeds** when α obtain a local argument A for L. A cooperative process **finishes** when there are no more replies from agents in $\mathcal{C}ooperate_\alpha$. A cooperative process **fails** when the cooperative process finished and the agent α does not built any local complete argument for L.

  - On receipt of an **ask($\gamma, L, player$)**: γ is sign as the *asking agent* for L. If α has only a local partial argument PA for L then α starts a **cooperative process** for PA, as $player$, by sending the message $ask(\alpha, L_i, player)$ for every objective literal $L_i \in Un(L, PA)$ to every $\beta \in \mathcal{C}ooperate_\alpha$. When α has a complete local argument A for L then α starts an **argumentative process** for A, as $player$, by sending to every $\beta \in \mathcal{A}rgue_\alpha$ the message $propose(\alpha, L, A, player)$.
  - On receipt of a **reply($\beta^c, L, Seq_L, player$)**: The agent α tries to complete the local partial argument PA (for L'), i.e. $A \in plus(PA, Seq_L)$. If the cooperative process succeeds, α starts an **argumentative process** for A, as $player$, by sending the message $propose(\alpha, L', A, player)$ to every $\beta^a \in \mathcal{A}rgue_\alpha$. If the cooperative process fails, α sends to the *asking agent* a $reply(\alpha, L', none, player)$; otherwise, α waits for other replies.
  - On receipt of a **reply($\beta^c, L, none, player$)**: If the cooperative process fails, α sends to the *asking agent* a $reply(\alpha, L, none, player)$; otherwise, α waits for other replies.

- **Argumentative process (for a complete argument A in α as player)**

  An argumentative process **succeeds** when the argument A is $acceptable_{p,o}$ w.r.t. $\mathcal{A}rgue_\alpha$. An argumentative process **inishes** when there are no more answers from agents in $\mathcal{A}rgue_\alpha$. An argumentative process **fails** when the argumentative process finished and the argument A is not $acceptable_{p,o}$ w.r.t. $\mathcal{A}rgue_\alpha$.

  - On receipt a **propose(γ, L, A)**: γ is sign as *proposing agent* of A, and $kp = Kp(o)$. For every $L_i \in Atts(A, p)$: if α has only a local kp-partial argument PA for L_i then α starts a **cooperative process** for PA, as opponent, by sending the message $ask(\alpha, L_i, o)$ to every $\beta^c \in$

$\mathcal{C}ooperate_\alpha$. When the cooperative process succeeds for PA (and so α has a argument CA for L_i that attacks A) then α sends to the proposing agent an $oppose(\alpha, L_i, CA)$. If the argumentative process fails, α sends to the proposing agent an $agree(\alpha, L, A)$.

- On receipt of an **oppose**(γ, L, A): γ is sign as *opposing agent* of A, and $kp = Kp(p)$. For every $L_i \in Atts(A, o)$: if α has only a local kp-partial argument PA for L_i then α starts a **cooperative process** for PA, as proponent, by sending the message $ask(\alpha, L_i, p)$ to every $\beta^c \in \mathcal{C}ooperate_\alpha$. When the cooperative process succeeds for PA (and so α has an argument CA for L_i that attacks A) then α sends to the opposing agent a $propose(\alpha, L_i, CA)$. If the argumentative process fails, α sends to the opposing agent an $agree(\alpha, L, A)$.
- On receipt an **agree**(γ, L, A): If the argumentative process fails, α sends to the proposing agent an $agree(\alpha, L, A)$; otherwise, waits for other answers.

As already said, a negotiation in a multi-agent setting $\mathcal{A} = \{Ag_1, \ldots, Ag_n\}$ is seen as a forest $\mathcal{F}$ of trees, where each involved agent α_i $(1 \leq i \leq n)$ has its own forest of trees $\mathcal{F}_i$. Therefore, $\mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_n$. Given the relation of our semantics with the well-founded semantics, it is natural that the structure of these forests are chosen to be similar to that of proof-procedures for the well-founded semantics of normal logic program, that rely on tabling techniques [Swi99, APS04]. In these, the nodes of the trees are either *regular nodes* of the form

$$NegotiableLiteral : -DelayList|LiteralList$$

or *failure nodes* of the form $fail$. A *Negotiable Literal* of an agent α for L as kp-player is a tuple $< \alpha, kp, L, Seq >$ such that $kp \in \{p, o\}$, and Seq is an available argument for L from α's set of available arguments (cf. Def. 59). Elements of $DelayList$ and $LiteralList$ are either objective or default literals. The literals in the $LiteralList$ are the ones that are yet left to be resolved in order to determine the truth value of the $NegotiableLiterals$. The $DelayList$ is used to deal with loops, in a way similar to that of tabling. This list contain literals whose evaluation has been delayed in order to avoid loops.

A negotiation of the truth value of an objective literal L started by an agent α is then represented by a tree T with root node $< \alpha, kp, L, [] >: -|L$ where $kp \in \{p, o\}$ indicates if α is a proponent or an opponent agent. We then say that T is a kp tree for L of α. The *negotiable literal* reflects no bindings to L then Seq is an empty argument for it, the $DelayList$ is empty, and the $LiteralList$ is the corresponding literal. Therefore, a single kp tree T in α sets up the negotiation process for L where agent α is a $kp-$player.

To evaluate the truth value of the literal L in a $NegotiableLiteral$ we must evaluate all literals in both $DelayList$ and $LiteralList$. The evaluation consists of starting either an argumentative or a cooperative dialogue for every literal in $LiteralList$. When a literal is selected that has been called before, we are in presence of a potential loop. In this case, as in tabling procedures, the literal is delayed (and added to the $DelayList$). Delayed literals can then either be solved with solutions obtained in the corresponding tree with that literal as root, and remain delayed until it is determined that no more solutions exist for that literals. This determination can be obtained by so called "table completion algorithms" that are well know for tabling procedures in logic programming [Swi99]. We represent a delayed literal as a tuple $< \beta, L >$ such that β is the argumentative (resp. cooperative) agent that will evaluate the truth value of the default literal (resp. build an argument for the objective) L.

During the negotiation process, the agent α receives either opposes or agrees (resp. replies) for each default (resp. objective) literal proposed (resp. asked) to $\mathcal{A}rgue_\alpha$ (resp. $\mathcal{C}ooperate_\alpha$). In such cases, the answers will be evaluated and the corresponding delay literal may be removed from $DelayList$. A regular leaf node N with an empty $LiteralList$ means that all literals were evaluated, and such a node is called an answer. If the $DelayList$ of N is also empty, meaning that all delayed literals were evaluated, N is called an *unconditional answer*; otherwise, N is called a *conditional answer*. In case of being a conditional answer, each tuple $< \beta, L >$ on $DelayList$ indicates that the argumentative (resp. cooperative) agent β does not agree with (resp. reply) L and so the next node will be a failure node.

On the implementation of the system

For implementing a system for the above sketched procedures, we propose an architecture which is based upon on three toolkits, viz. *JGroups* [Bea02], *Interprolog* [Cal04], and *XSB Prolog* [SW07]. Since we have two kinds of groups in a multi-agent setting, viz. argumentative agents and cooperative agents, the corresponding argumentation-based negotiation system should deal with the creation and the control of both groups and their membership. Furthermore, the exchange of messages between either argumentative and cooperative agents should be reliable. We choose *JGroups* [Bea02], a toolkit for building reliable group communication for Java applications. JGroups permits agents' communication in Local Area Networks (LAN) or Wide Area Networks (WAN), the delivery of messages to every involved agent. New agents or crashed agents are handled respectively in the sets of argumentative and cooperative agents. We implement the argumentation-based negotiation framework in *XSB Prolog* [SW07]. XSB is an open source logic programming system that extends Prolog with new semantic and operational features, mostly based on the use of Tabled Logic Programming

or tabling, see e.g. [CW96, Swi99]. The use of such a resource permits, during a computation of a goal G, where $G \leftarrow Body$ is a rule in a logic program, each 'subgoal' $S \in Body$ to be registered in a table the first time it is called, and unique answers to S to be added to the table as they are derived. When subsequent calls are made to S, the evaluation ensures that answers on S are read from the table rather than being re-derived using program clauses. Thus, a first advantage of tabling is that it provides termination to a logic program; more details about such a toolkit will be presented in the remainder of this chapter. Since Jgroups is implemented in Java language, we need an intermediate level between JGroups and XSB, to map Java objects to Prolog terms (and vice-versa). *Interprolog* [Cal04] is a toolkit for developing Java + Prolog applications, which provides an application program interface (API) for directly mapping Java objects to Prolog terms. Furthermore, Interprolog supports a Prolog process through the API.

Therefore, the proposed architecture for the implementation of the argumentation-based negotiation system is composed of a network communication layer and an inference engine. Figure 4.6 illustrates such an architecture. The *network communication layer* is based on the toolkit JGroups for building reliable group communication. The *inference engine* is based on the toolkit Interprolog, the middleware for Java and Prolog, which provides method/predicate calling between both. Finally, the *goal-interpreter* is the XSB System, which computes the argumentation-based negotiation's Prolog implementation over an agent's knowledge base. In the remainder of this section we provide some details on the communication layer, and its connection to the XSB inference engine.

A Network Communication Layer As already said, since the argumentation-based negotiation system should deal with both groups of argumentative and cooperative agents in a multi-agent setting, the network communication layer should guarantee both creation and control of such groups. Furthermore, the exchange of messages between group members should be reliable. We therefore propose a communication network layer by using the toolkit JGroups. JGroups is a reliable group communication toolkit written entirely in Java. It is based on IP multicast, but extends it with reliability and group membership. *Reliability* includes (among other things): (1) lossless transmission of a message to all recipients (with retransmission of missing messages); (2) fragmentation of large messages into smaller ones and reassemble at the receiver's side; (3) ordering of messages (FIFO order); (4) atomicity, i.e. a message will be received by all receivers or none. *Group Membership* includes (1) knowledge of who the members of a group are, and (2) notification when a new member joins, an existing member leaves, or an existing member has crashed

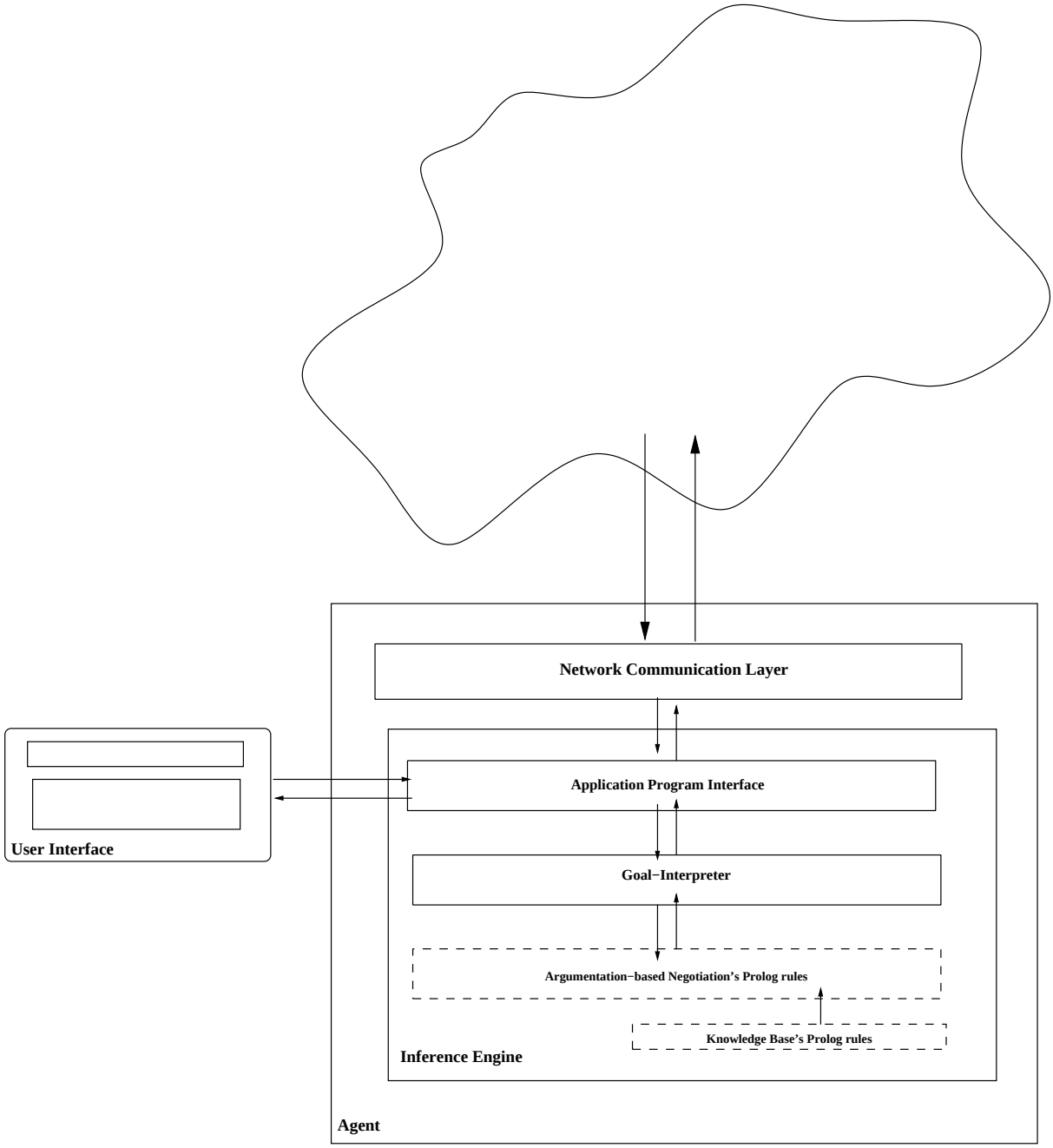


Figure 4.6: An Architecture for Argumentation-based Negotiation

	Unreliable	**Reliable**
Unicast	UDP	TCP
Multicast	IP Multicast	*JGroups*

In unicast communication, where one sender sends a message to one receiver, there is UDP and TCP. UDP is unreliable, packets may get lost, duplicated, may arrive out of order, and there is a maximum packet size restriction. TCP is also unicast, but takes care of message retransmission for missing messages, weeds out duplicates, fragments packets that are too big and present messages to the application in the order in which they were sent. In the multicast case, where one sender sends a message to many receivers, IP Multicast extends UDP: a sender sends messages to a multicast address and the receivers have to join that multicast address to receive them. Like in UDP, message transmission is still unreliable, and there is no notion of membership (who has currently joined the multicast address).

JGroups extends reliable unicast message transmission (like in TCP) to multicast settings. As described above it provides reliability and group membership on top of IP Multicast. Since every application has different reliability needs, JGroups provides a flexible *protocol stack*, which allows developers to adapt it to match their application requirements and network characteristics exactly. Furthermore, by mixing and matching the available protocols of JGroups, the requirements of our argumentation-based negotiation system are satisfied. Moreover, since protocols are independent of each other, they can be modified, replaced or new ones can be added, improving the modularity and maintainability of our system. The chosen protocol stack for the argumentation-based negotiation system is as follows:

- the transport protocol uses TCP (from TCP/IP) to send unicast and multicast messages. In the latter case, each message to the group is sent as multiple unicast messages (one to each member). Due to the fact that IP multicasting cannot be used to discover the initial members, another mechanism has to be used to find the initial membership. The TCPPING uses a list of well-known group members that TCP solicits for initial membership. So, TCPPING determines the initial membership and denotes it as the coordinator of the group. Every request to join will then be sent to the coordinator;

- to add loss-less transmission, we choose the "Negative Acknowledgement Layer" protocol (NACKAC). NACKAC ensures message reliability and "First In First Out"(FIFO). Message reliability guarantees that a message will be received. If not, the receiver will request retransmission. FIFO guarantees that all messages from sender P will be received in the order P sent them;

- the "Group Membership Service" (GMS) provides for group membership.

It allows the system to register a callback function that will be invoked whenever the membership changes, e.g. a member joins, leaves or crashes;

- the "Failure Detector" (FD) and the "Distributed Message Garbage Collection" (STABLE) are needed by the GMS to announce crashed members.

The API of JGroups is very simple and is always the same, regardless of how the underlying protocol stack is composed. To send/receive messages, an instance of *Channel* has to be created. Channels are the means by which member processes in a group can communicate with each other; they are used to send and receive messages and views of a group. A channel represents a single member process; at any point only a single process can be connected to a channel. When a client connects to a channel, it indicates the group address (usually the name of the group) it would like to join. Thus, a connected channel is always associated with a particular group. The GMS takes care that channels with the same group name find each other: whenever a client, given a group name, connects to a channel, then the GMS tries to find existing channels with the same group name and joins them; the result is a new view being installed (which contains the new member).

The channel's properties are specified when creating it, and this causes the creation of an underlying protocol stack. All protocol instances are kept in a linked list (i.e. the protocol stack), where messages move up/down. The reliability of a channel is specified as a string; we choose the transport protocol TCP with TCPPING and so the channel's properties should look like as follows:

```
"TCP(start_port=7800):" +
"TCPPING(initial_hosts=HostA[7800],HostB[7800];port_range=5;"+
"timeout=5000;num_initial_members=2;up_thread=true;"+
"down_thread=true):" + ...
```

which means that HostA and HostB are designated members that will be used by TCPPING to look up the initial membership. The property start_port in TCP means that each member should try to assign port 7800 to itself. If this is not possible, it will try the next higher port (7801) and so on, until it finds an unused port. TCPPING will try to contact both HostA and HostB, starting at port 7800 and ending at port 7800 + port_range (in the above example, it means ports 7800 - 7804). Assuming that at least HostA or HostB is up, a response will be received. To be absolutely sure to receive a response, all the hosts on which members of the group will be running should be added to the configuration string.

The state transition of a channel is described in the following. When a channel is first created, it is in an unconnected state. An attempt to perform certain operations which are only valid in a connected state (e.g. send/receive messages) will result in an exception. After a successful connection by a client, the channel

status moves to a connected state. Then channels will receive messages, views and suspicions from other members; they may also send messages to other members or to the group (getting the local address of a channel is guaranteed to be a valid operation in this state). When the channel is disconnected, it moves back to an unconnected state. Both connected and unconnected channel may be closed, which makes the channel unusable for further operations. Any attempt at closure will result in an exception. When the channel is closed directly from a connected state, it will first be disconnected, and then closed. The basic methods to use a channel is presented in the following:

To join a group, the method

```
public void connect(String groupName) throws ChannelClosed
```

should be called. It returns when the member has successfully joined the group, or when it has created a new group (if it is the first member, it is denoted as coordinator of such a group).

Then a message is sent using the method

```
public void send(Message msg) throws ChannelNotConnected,
                                     ChannelClosed
```

such that *msg* contains a destination address, a source address and a byte buffer. The destination should be either an address of the receiver (unicast) or *null* (multicast). When it is *null*, the message will be sent to all members of the group (including itself[12]). If the source address is *null*, it will be set to the channel's address, and so every recipient may generate a response and send it back to the sender. A String object is set to be the message's contents and it is serialised into the *msg*'s byte buffer.

A channel receives messages asynchronously from the network and stores them in a queue. When the method

```
public Object receive(long timeout) throws ChannelNotConnected,
                                           ChannelClosed, Timeout
```

is called, the next available message from the top of the queue is removed and returned. When there are no messages in the queue, the method will be blocked. If timeout is greater than 0, it will wait the specified number of milliseconds for a message to be received, and throw a Timeout exception if none is received during that time. If the timeout is 0 or negative, the method will wait indefinitely for the next available message.

The member disconnects from the channel by using the method

[12]Unless the channel option LOCAL is set to 'false'.

```
public void disconnect()
```

and the consequence is that the channel removes itself from the group membership. This is done by sending a leave request to the current coordinator (e.g. the first member of the group). The coordinator will subsequently remove the channel's address from its local view, and send the new view to all remaining members of the group.

Finally, to destroy a channel instance, the method

```
public void close()
```

is used. It moves the channel to a closed state, in which no further operations are allowed.

The following code illustrates how to use such methods:

```
String props="UDP:FD:NAKACK:STABLE:GMS";
Message send_msg;
Object  recv_msg;

Channel channel=new JChannel(props);
channel.connect("MyGroup");

send_msg=new Message(null, null, "Hello World");
channel.send(send_msg);

recv_msg=channel.receive(0);
System.out.println("Received " + recv_msg);

channel.disconnect();
channel.close();
```

Channel provides asynchronous message sending/reception, somewhat similar to UDP. A message sent is essentially put on the network and the method *send()* will return immediately. Conceptual requests, or responses to previous requests, are received in an undefined order, and the application has to take care of matching responses with requests. Also, an application has to retrieve messages actively from a channel (pull-style), i.e. it is not notified when a message has been received. Note that pull-style message reception often needs some form of event-loop, in which a channel is periodically polled for messages. However, JGroups has a *building blocks package* which provides more sophisticated APIs on top of a channel. Building blocks either create and use channels internally, or require an existing channel to be specified when creating a building block. Applications communicate directly

with a building block, rather than with a channel. Therefore, an application programmer does not need to write, for instance, a request-response correlation.

The *PullPushAdapter* belongs to the building blocks package, and it is a converter between the pull-style of actively receiving messages from the channel and the push-style where clients register a callback, which is invoked whenever a message has been received. Clients of a channel do not have to allocate a separate thread for message reception, they have to implement the *interface MessageListener*:

```
public interface MessageListener {
  public void receive(Message msg);
  byte[] getState();
  void setState(byte[] state);
}
```

whose method *receive()* will be called when a message arrives. Both methods *getState()* and *setState()* are used to fetch and set the group state (e.g. when joining). Furthermore, clients interested in being called when a message is received should register with the PullPushAdapter using the method

```
public void registerListener(java.io.Serializable identifier,
                             MessageListener l)
```

which sets a listener to messages with a given identifier; through this identifier, the header will be routed to this listener. The *MembershipListener interface* is similar to the above interface: every time a new view, a suspicion message, or a block event is received, the corresponding method of the class implementing MembershipListener will be called.

```
public interface MembershipListener {
    public void viewAccepted(Viewnew_view);
    public void suspect(Object suspected_mbr);
    public void block();
}
```

When a client is interested in getting a view, suspicion messages and blocks, then it must additionally register as a *MembershipListener* using the method

```
public void addMembershipListener(MessageListener l)
```

and whenever a view, suspicion or block is received, the corresponding method will be called. Often, the only method containing any functionality will be *viewAccepted()*, which notifies the receiver that a new member has joined the group

or that an existing member has left or crashed. The *suspect()* callback is invoked by JGroups whenever a member is suspected of having crashed. The block method is called whenever the member needs to stop sending messages[13]. Figure 4.7 illustrates how pullPushAdapter runs with both application and channel. It constantly pulls messages from the channel and forwards them to the registered listeners. Thus, an application does not have to actively pull for messages; the PullPushAdapter does this for it. However, the application has to access the channel directly if it wants to send a message.

Connection to the Inference Engine Interprolog is a middleware for Java and Prolog, providing method/predicate calling between both. Interprolog's innovation is its mapping between (serialized) Java Objects and their Prolog specifications, propelled by the Java Serialization API which does most of the work of the Java side. The Prolog side is built upon a "Definite Clause Grammar"(DCG)[14] that analyses/generates (the bytes of) serialized objects (Figure 4.8 illustrates this). Furthermore, InterProlog supports multiple Prolog threads and it is compatible with ISO Prolog. Despite having such a feature, the proposed architecture for argumentation-based negotiation has only one goal-interpreter and so only one thread is needed.

Interprolog defines that each Prolog System has a specific *PrologEngine* where most system-dependent knowledge is. The *PrologEngine* is the 'heart' of InterProlog, it represents a Prolog machine instance. Since PrologEngine is an abstract class, we should choose one of its subclasses. *NativeEngine* is a XSB Prolog engine implemented using Java Native Interface; this subclass depends on the XSB Prolog package, and so the file path to the directory containing the Prolog binary must be specified. The following fragment allows a Java Progammer to use XSB Prolog, consult a Prolog file, and perform a simple query:

```
PrologEngine engine = new NativeEngine( xsbDir );
if( !engine.command("["+file+"]") ){
   System.out.println( "ERROR: to read the Prolog program "+file );
} else try{
     Object[] bindings =
        engine.deterministicGoal(''father(X,john)'',''[string(X)]'');
     if( bindings!=null ){
```

[13]The *block()* callback is only needed by the Virtual Synchrony suite of protocols; otherwise, it will never be invoked. For details see [Bea02].

[14]DCGs are a special notation provided in most Prolog systems which provide a convenient way of defining grammar rules. The general form of each DCG clause is $Head --> Body$ meaning "a possible form for $Head$ is $Body$" such that both $Head$ and $Body$ are Prolog terms and $-->$ is an infix operator.

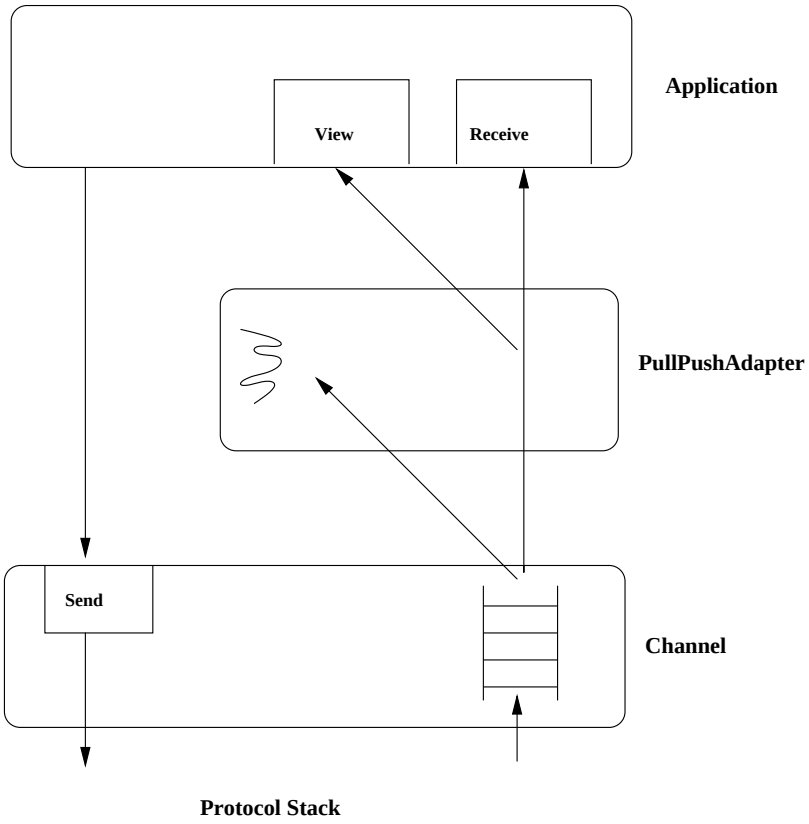


Figure 4.7: PullPushAdapter

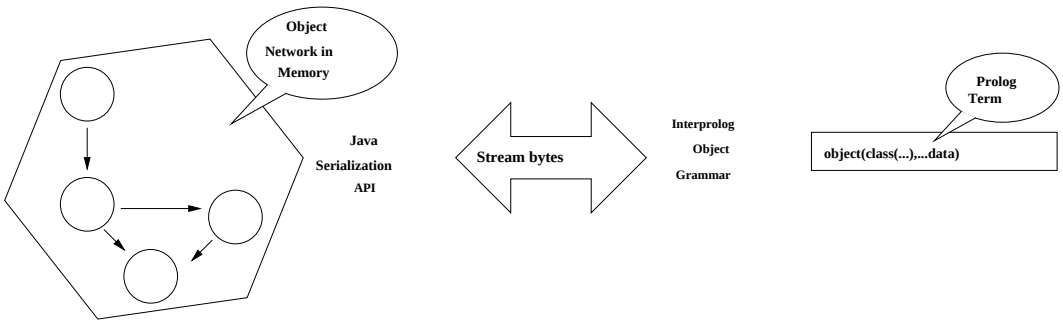


Figure 4.8: Interprolog as a middleware for Java and Prolog

```
    String X = (String) bindings[0];
    System.out.println(''The father of john is ''+ X );
}
```

The method *command()* executes a Prolog predicate with no other result than success/failure and the method *deterministicGoal()* is useful when we are constructing objects from Prolog, but do not need to pass any information to Java.

Chapter 5

Related Work

According to *The Uses of Computational Argumentation* (2009), argumentation is a form of reasoning in which explicit attention is paid to the reasons for the conclusions that are drawn and how conflicts between reasons are resolved. Explicit consideration of the support for conclusions provides a mechanism, for example, to handle inconsistent and uncertain information. Argumentation has been studied both at the logical level, as a way of modelling defeasible inference, and at the dialogical level, as a form of agent interaction. Argumentation has long been studied in disciplines such as philosophy, and one can find approaches in computer science from the 1970s onwards that clearly owe something to the notion of an argument. Work on computational argumentation, where arguments are explicitly constructed and compared as a means of solving problems on a computer, first started appearing in the second half of the 1980s, and argumentation is now well established as an important sub-field within Artificial Intelligence. According to *Argument, Dialog and Decision* (2010), since the work of John Pollock, Ronald Loui and others in the eighties, argumentation has proven to be successful in nonmonotonic logic. In the nineties, Dung and others showed that argumentation is also very suitable as a general framework for relating different nonmonotonic logics. Finally, in recent years argument-based logics have been used to facilitate reasoning and communication in multi-agent systems. Argumentation can be studied on its own, but it also has interesting relations with other topics, such as dialogue and decision. For instance, argumentation is an essential component of such phenomena as fact finding investigations, computer supported collaborative work, negotiation, legal procedure, and online dispute mediation.

In the following we relate our work with proposals in semantics of abstract argumentation systems, defeasible reasoning, and argumentation-based negotiation. The section *Semantics of Abstract Argumentation Systems* evaluates some extended-based semantics presented by a recent overview in [BG09]. The section *Defeasible Reasoning* is related with work presented in Chapter 3. This section

provides complementary information of Chapter 2, and our intention is to evaluate approaches presented by [GDS09], a recent overview of Defeasible Reasoning and Logic Programming. The last section, *Argumentation-based Negotiation*, is related with work presented in Chapter 4 and we focus our attention in preference-based argumentation frameworks.

5.1 Semantics of Abstract Argument Systems

An *abstract argumentation system* or *argumentation framework*, as introduced by [Dun95], is simply a pair $< \mathcal{A}, \mathcal{R} >$ consisting of a set $\mathcal{A}$ whose elements are called *arguments* and of a binary relation $\mathcal{R}$ on $\mathcal{A}$ called *attack relation*. Moreover, an abstract argument is not assumed to have any specific structure but it is anything that may attack or be attacked by another argument. Similarly, the attack relation has no specific meaning: if an argument b attacks another argument a, denoted by (A_b, A_a), this means that if b holds then a cannot hold. Furthermore, given that arguments may attack each other, it is clear that they cannot stand all together and their status is subject to evaluation. The evaluation process (in abstract argument systems) concerns the justification state of arguments. Intuitively, an argument is justified if it has some way to "survive" the attacks it receives; it is not justified (it is rejected), otherwise.

An *argumentation semantics* is the formal definition of a method (either declarative or operational) ruling the argument evaluation process. Two main styles of argumentation semantics definition can be identified in the literature: *extension-based* and *labelling-based*. In an extension-based approach a semantics definition specifies how to derive from an argumentation framework a set of *extensions*. An extension E of an argumentation framework $< \mathcal{A}, \mathcal{R} >$ is simply a subset of $\mathcal{A}$ representing a set of arguments which are "collectively acceptable". In a labelling-based approach a semantics definition specifies how to derive from an argumentation framework a set of *labellings*. A labelling $\mathcal{L}$ is the assignment to each argument in $\mathcal{A}$ of a label taken from a predefined set $\mathcal{L}$, which corresponds to the possible alternative states of an argument in the contex of a single labelling.

[BG09] shows that for a given argumentation framework one or more extensions (labellings) may be prescribed by a given semantics. If a semantics S always prescribes exactly one extension (labelling) for any argumentation framework (where the semantics is defined) then S is said to belong to the *unique-status* (or *single-status*) approach; otherwise, it is said to belong to the *multiple status* approach. Furthermore, from an historical point of view, [BG09] distinguishes several extension-based argumentation semantics as follows:

- Four "traditional" semantics, considered in [Dun95]'s argumentation proposal, namely *complete*, *grounded*, *stable*, and *preferred* semantics. An over-

view of them is done in Chapter 2.

- Subsequent proposals introduced by various authors in the literature, often to overcome some limitations or improve some undesired behavior of a traditional approach, namely *stage*, *semi-stable*, *ideal*, *CF2*, and *prudent* semantics.

Grounded, *ideal*, and *prudent* semantics belong to the unique-status approach. Since our proposal follows the unique-status approach we will focus our attention on these semantics.

Grounded Semantics As already said, we follow the grounded semantics. Our results and comparison with such this proposal are presented in Chapter 3. There, we relate our results to the *grounded (skeptical) extension* (cf. Theorem 36).

Ideal Semantics The ideal semantics [DMT06] allows the acceptance of a set of arguments possibly larger than is case with grounded semantics, as shown by Example 39 (adapted from [BG09]). First, the definition of the semantics is presented:

Definition 68 (Ideal Semantics) *Let* $AF =< \mathcal{A}, \mathcal{R} >$ *be an argumentation framework,* $E_{\mathcal{PR}}(AF)$ *be a set of preferred extensions*[1] *of* AF*, and* $S \subseteq \mathcal{A}$*. S is an* ideal set *iff S is admissible*[2] *and* $\forall E \in E_{\mathcal{PR}}(AF)\ S \subseteq E$*. The* ideal extension *is the maximal (w.r.t. set inclusion) ideal set.*

Example 39 *Let* $AF =< \mathcal{A}, \mathcal{R} >$ *be an argumentation framework such that*

$$\mathcal{A} = \{A_a, A_b, A_c, A_d\}$$
$$\mathcal{R} = \{(A_a, A_b), (A_b, A_a), (A_a, A_c), (A_b, A_c), (A_c, A_d), (A_d, A_c)\}$$

The grounded extension is $\emptyset$*. Since* $E_{\mathcal{PR}}(AF) = \{\{A_a, A_d\}, \{A_b, A_d\}\}$*, the ideal extension is* $\{A_d\}$*. As already proved, our (centralized) argumentation framework has the same results as the grounded semantics.*

We then adapt the above AF to $\mathcal{A} = \{Ag_1, Ag_2\}$ *such that*

$$Ag_1 = < 1, \{a \leftarrow not\ b; b \leftarrow not\ a\}, \{1\}, \{1, 2\} >$$
$$Ag_2 = < 2, \{c \leftarrow not\ a; c \leftarrow not\ b; c \leftarrow not\ d; d \leftarrow not\ c\}, \{2\}, \{1, 2\} >$$

[1] A preferred extension of A is as large as possible and able to defend itself from attack. For further details see Definition 16 on Section 2.2.

[2] For details see Definition 14 on Section 2.2.

such that Ag_1 cooperates with Ag_2 (and vice versa), and the arguments that attack each other are in the same agent. We show that any other distribution of such arguments has the same result for this "configuration" of $\mathcal{A}$. We defined that, for a given agent α in a multi-agent setting $\mathcal{A}$, an agent $\beta \in \mathcal{C}ooperate_\alpha$ cooperates with an available argument A (cf. Def.59) under one of the following conditions: (i) A is not attacked by any argument from $\mathcal{A}rgue_\beta$, or (ii) A is attacked, but every attacking argument (cf. Def.60) B against A is attacked by some argument from $\mathcal{A}rgue_\beta$. In both cases, A is considered a defensible argument *(cf. Def.62). Besides of the fact the arguments for a, b, c, and d are involved in "mutual attack", those arguments are used in a cooperation process. Then, the arguments for a, b, c, and d are $defensible^{s,k}_{\mathcal{A}}$ ($k \in \{s, w\}$) in both agents.*

Therefore, the ideal semantics allows the acceptance of a set of arguments possibly larger that our (distributed) argumentation-based negotiation when every agent cooperates with all agents in $\mathcal{A}$. However, we obtain different results for any other configuration of sets of argumentative and cooperative agents. In the conclusion of this chapter we present such configurations.

Prudent Semantics The family of prudent semantics [CMDM05] is introduced by considering a more extensive notion of attack in the context of traditional semantics. In particular, an argument A_a *indirectly attacks* an argument A_b if there is an odd-length attack path from A_a to A_b. The odd-length path does not need to be the shortest path and include cycles. A set S of arguments is free of indirect conflicts, denoted by $icf(S)$, if $\nexists A_a, A_b \in S$ such that A_a indirectly attacks A_b. The prudent version of traditional grounded extension is defined as follows:

Definition 69 *Let $AF =< \mathcal{A}, \mathcal{R} >$ be an argumentation framework. The function $F^p_{AF} : 2^{\mathcal{A}} \rightarrow 2^{\mathcal{A}}$ such that for a given set $S \subseteq \mathcal{A}$, $F^p_{AF}(S) = \{A_a \mid A_a$ is acceptable w.r.t. $S \wedge icf(S \cup \{A_a\})\}$ is called a p(rudent)-characteristic function of AF. Let j be the lowest integer such that the sequence $(F^p_{AF}(S))^i(\emptyset)$ is stationary for $i \geq j : (F^p_{AF}(S))^i(\emptyset)$ is the* grounded p(rudent)-extension *of AF, denoted as $GPE(AF)$.*

As $F^p_{\mathcal{AF}}$ is more restrictive than F_{AF} [Dun95], it follows that the prudent version of grounded semantics ($\mathcal{GRP}$) is a possible strict subset of the traditional grounded semantics ($\mathcal{GE}$). This entails in particular that reinstatement is given up by $\mathcal{GRP}$ as it can be seen in the following example (adapted from [BG09]):

Example 40 *Let* $AF =< \mathcal{A}, \mathcal{R} >$ *be an argumentation framework such that* $\mathcal{A} = \{A_a, A_b, A_c, A_d, A_e\}$ *and* $\mathcal{R} = \{(A_a, A_b), (A_b, A_c), (A_e, A_c), (A_c, A_d)\}$. $GE(AF) = \{A_a, A_d, A_e\}$ *whereas* $GPE(AF) = \{A_a, A_e\}$ *because* A_d *is not reinstated.*

We do not deal with 'indirect attack', and so $F^p_{\mathcal{AF}}$ is also more restrictive than our characteristic function. Since we propose a distributed version of argumentation, every agent would have to control the 'odd-length attack path' to obtain similar results to centralized prudent semantics.

5.2 Defeasible Reasoning

According to [GS04], research in Nonmonotonic Reasoning, Logic Programming, and Argumentation has provided important results seeking to develop more powerful tools for knowledge representation and common sense reasoning. Advances in those areas are leading to important and useful results for other areas such as the development of intelligent agents and multi-agent system applications. Therefore,

> "Defeasible Argumentation is a relatively young area, but already mature enough to provide solutions for other areas. Argumentative systems are now being applied for developing applications in legal systems, negotiation among agents, decision making, etc"[GS04]

Nute was the first to introduce the *Logic for Defeasible Reasoning* [Nut94] (LDR), a formalism that provides defeasible reasoning with a simple representational language. Although LDR is not a defeasible argumentation formalism in itself, its implementation — d-Prolog — defined as an extension of PROLOG, was the first language that introduced defeasible reasoning programming with specificity as a comparison criterion between rules. The proposed language has no default literals, only a literal and its strong negation[3]. In LDR there are three types of rules: strict rules (e.g. *emus are birds*), defeasible rules (e.g. *birds usually fly*) and defeater rules (e.g. *heavy animals may not fly*). The purpose of defeater rules is to account for the exceptions to defeasible rules and so they can only be used to block an application of a defeasible rule. LDR also has defined strict and defeasible derivations. The former only uses facts and strict rules; in the latter, a more sophisticated analysis is performed such that if a literal has a defeasible derivation then no further analysis is performed. A defeasible derivation cannot

[3]The strong negation represents contradictory knowledge, and it was introduced by [GL90]. It is also known as "explicit negation".

be considered as a single argument because it is related to a tree of arguments that encodes the analysis of all possible attacks and counter attacks.

Pollock's proposal [Pol74, Pol87] is another important work in Defeasible Reasoning. It introduced an important distinction between two kind of defeat, namely rebutting defeat (attack on a conclusion) and undercutting defeat (attack on an inference rule). In [Pol95], Pollock has changed the way in which an argument is warranted[4] adopting a *multiple status assignment approach*[5]. Pollock has also developted a computer program in LISP — OSCAR [Pol96] — where arguments are sequences of linked reasons, and probabilities are used for comparing competing arguments. In a way similar to Nute's defeater rules, explicit undercutting defeaters can be expressed in his language. Differently from Nute's proposal, an inference graph is used by OSCAR for evaluating the status of arguments.

Dung states that "there are interesting relations between argumentation and logic programming"[Dun95], and he shows that logic programming can be shown as a particular form of argumentation. He shows that argumentation can be "viewed" as logic programming by introducing a general method for generating an interpreter for argumentation. This method consists of a very simple logic program consisting of the following two clauses:

$$acceptable(A) \leftarrow not\ \ defeated(A).$$
$$defeated(A) \leftarrow attacks(A), acceptable(A).$$

Inspired by legal reasoning, Prakken and Sartor [PS97] introduce an argument-based formalism for *extended logic programming with defeasible priorities*, designed as an instance of Dung's abstract argumentation framework with Grounded semantics. In their formalism, arguments are expressed with both strong and default negation. Conflicts between arguments are decided with the help of priorities on the rules. These priorities can be defeasible derived as conclusions within the system. Its declarative semantics is given by a fixed point definition. Since they are inspired by legal reasoning, a proof of a formula takes the form of a dialogue between a proponent and an opponent: an argument is shown to be justified if the proponent can make the opponent run out of moves in whatever way the opponent attacks. Proponent and opponent have different rules for introducing arguments, leading to an asymmetric dialogue. As already said, we follow Dung's and Prakken and Sartor's proposal. An overview of these proposals is done in Section 2.3.2, and our results and a comparison are presented in Chapter 3. Then:

- We generalize [PS97]'s definition of argument by proposing two kinds of arguments, viz. strong arguments and weak arguments. Having two kinds of

[4]The terminology varies in the literature: *justified* vs *warranted*, *overruled* vs *defeated*, *defensible*, etc.

[5]Unique- and multiple-status assignments for arguments are presented in Chapter 2, and analyzed in depth in [PV02].

arguments, attacks by rebut do not need to be considered. Simply note that rebut is undercut against weak arguments. Therefore, rebut is not considered in our proposal since, as already shown in [SdAMA97, dAMA98b, SS02b], it can be reduced to undercut by considering weaker versions of arguments.

- We extend [PS97]'s argumentation-based semantics for *extended logic programs* to deal with denials.

- Similarly to [Dun95, PS97] we formalize the concept of acceptable arguments with a fixpoint operator. However, the acceptability of an argument may have different results and it depends on which kind of interaction between (strong and weak) arguments is chosen. Therefore, our argumentation semantics assigns different levels of acceptability to an argument and so it can be justified, overruled, or defensible. Moreover, a justified argument can be contradictory, based on contradiction, or non contradictory. Consequently, the truth value of an literal can be true (and either contradictory, based on contradiction, or non contradictory), false, or undefined.

Garcia's proposal of Defeasible Logic Programming [Gar00] (DeLP), is a formalism which combines results from Logic Programming and Defeasible Argumentation. DeLP provides the possibility of representing information in the form of "weak rules"[6] in a declarative manner, and a defeasible argumentation inference mechanism for warranting the entailed conclusions. DeLP considers two kinds of program rules, viz. defeasible rules and strict rules. A *defeasible rule* express that *reasons to believe in the antecedent (Body) provide reasons to believe in the consequent (Head)*, and it is used as a tentative information that may be used if nothing could be posed against it; and a *strict rule* is used to represent non-defeasible information. Therefore, defeasible rules are used for representing weak or tentative information, like "a mammal does not fly" or "usually, a bird can fly"; and strict rules[7] are used for representing strict (sound) knowledge, like "a dog is a mammal" or "all penguins are birds"

The DeLP language is defined in terms of three disjoint sets: a set of facts, a set of strict rules, and a set of defeasible rules; and both strict and defeasible rules are ground. A DeLP-program [GS04] is denoted by a pair (Π, Δ) such that Π is a set of facts and strict rules, and Δ is a set of defeasible rules. A derivation from $(\Pi, \emptyset)$ is called *strict derivation*; otherwise it is a *defeasible derivation.* A DeLP-query for a ground literal Q succeeds if it is possible to build an argument A that supports Q, and A is found to be undefeated by a warranted procedure. The *warranted procedure* evaluates if there are other arguments that counter-argue

[6]Weak rules were proposed by [Pol95] and they are used to represent relations between pieces of knowledge that could be defeated after all things are considered.

[7]Syntacticly, strict rules correspond to *basic rules*[Lif96].

or attack A (or a sub-argument of A). In order to verify whether an argument is non-defeated, all of its associated counter-arguments have to be verified, each of them being a potential reason for rejecting A. Then it is established an *argument comparison criterion* based on generalized specificity, e.g. a more "precise argument" is preferred.

DeLP accepts that (Π, Δ) can have contradictory information but it does not derive contradictory literals, i.e. neither a literal nor its strong negation are warranted. DeLP has four possible answers for a literal L: *yes*, *no*, *undecided* or *unknown*. The first means that L is *warranted*; the second means the strong negation of L is *warranted*; the third one is related with a contradiction detected between both L and its strong negation; and the last one is related with incomplete information about L. Moreover, every literal in the Body of a defeasible rule may be undecided if the Head is contradictory. Therefore, every literal involved with some kind of contradiction is *undecided*.

According to [Pra09], in both [Nut94]'s and [Gar00]'s proposals — respectively LDR and DeLP systems — the logic language is restricted in logic programming. LDR is not explicitly argument-based but defines the notion of a proof tree, which interleaves support and attack. LDR is proven to instantiate ground semantics [Dun95]. In DeLP the only way to attack an argument is on a (sub-) conclusion. DeLP's notion of argument acceptability has no known relation to any of the current argumentation semantics. Nevertheless, we relate our work with DeLP's proposal [Gar00] as follows:

- As in our proposal, no priority relation is needed for deciding between contradictory literals in DeLP. However, DeLP distinguishes strict and defeasible rules: only defeasible rules are evaluated. In some sense, it is a kind of 'weak' preference. The definition of a formal criterion for comparing arguments is a central problem in defeasible argumentation. Existing formalisms have adopted different solutions. Abstract argumentation systems usually assume an ordering in the set of all possible arguments (eg. [Dun93]). In other formalisms, explicit priorities among rules are given, so that the conflict between two rules can be solved. This approach is used in [Nut94]. In [PS97] it is also possible to reason (defeasibly) about priorities among rules. An alternative is to use the specificity criterion, and no explicit order among rules or arguments need to be given. Finally, other formalisms use no preferences [Gar00, GS04], as is also our case. However, it seems clear that the flexibility offered by our proposal of sets of cooperative and argumentative agents allows for giving priority to sets of rules over other sets of rules. This is somehow similar to what is done in preferences in the context of logic programs. We have illustrated it in examples of Section 4.5, viz. how to model knowledge over a hierarchy, and also how to model knowledge that evolves.

Both cases specify priorities between agents' sets of rules.

- Since our argumentation is parameterized by the kind of interaction between arguments, we obtain results ranging from a consistent way of reasoning to a paraconsistent way of reasoning; the former is more sceptical than the latter and it has similar results if applied to the DeLP's set of defeasible rules. The major difference between DeLP and our approach is the way in which contradictory conclusions are treated. DeLP concludes both contradictory literals from defeasible rules as *undecided*. We assume, in a consistent way, that every contradictory literal has to be $undefined_P^{w,w}$. However, in a paraconsistent way, we deal with contradiction and a literal may be true and contradictory, based on contradiction, or non contradictory (for details see Def. 49). So we may have contradictory literals as ($true_P^{s,w}$ and) $contradictory_P^{s,w}$. We further consider the literals that are involved with contradiction but in a different way, every literal that is both $true_P^{s,w}$ and $false_P^{s,w}$ (and not $contradictory_P^{s,w}$) is based-on-$contradiction_P^{s,w}$, whereas DeLP concludes them as *undecided*. $Undefined_P^{k,w}$ literals are only those which are neither $true_P^{k,w}$ nor $false_P^{k,w}$ ($k \in \{s, w\}$). For better understanding see Example 41.

Example 41 *[GDS09] presents the following DeLP-program* (Π, Δ) *where* Π *is a set of facts and strict rules, and* Δ *is a set of defeasible rules.*

$$
\begin{aligned}
\Pi = \{ \quad & switch\_on(a); switch\_on(b);\ \neg electricity(b); \\
& night;\ sunday;\ deadline; \\
& \neg day \leftarrow night; \\
& \neg dark(X) \leftarrow illuminated(X)\} \\
\Delta = \{ \quad & lights\_on(X) \leftarrow switch\_on(X); \\
& \neg lights\_on(X) \leftarrow \neg electricity(X); \\
& lights\_on(X) \leftarrow \neg electricity(X),\ emergency\_lights(X); \\
& dark(X) \leftarrow \neg day; \\
& illuminated(X) \leftarrow lights\_on(X), \neg day; \\
& working\_at(X) \leftarrow illuminated(X); \\
& \neg working\_at(X) \leftarrow sunday; \\
& working\_at(X) \leftarrow sunday, deadline\}
\end{aligned}
$$

Π *has information about two rooms, viz. a and b. There are facts expressing that in both rooms a and b the light switch is on, and in room b there is no electricity. There are also facts expressing that it is Sunday night and that people working there have a deadline. The last strict rule expresses that an illuminated room is not dark.* Δ *has the defeasible rules that can infer, for instance, which room is illuminated or if someone is working in a particular room. The first rule states that "reasons to believe that the light switch of a room is on, provide reasons to*

believe that the lights on that room are on". The second rule expresses that "usually if there is no electricity then lights of a room are not on". The third rule states that "normally, if there is no electricity but there are emergency lights, the lights will be on". The last two rules state that "normally there is nobody working in a room on a Sunday", however, "if they have a deadline, people may be working on Sunday".

In this example, the literals illuminated(a), working_at(a), ¬dark(a), and dark(b) are warranted. In [GDS09] it is shown that two contradictory literals trivially disagree. This is the case of working_at(a) and ¬working_at(a), however the defeasible derivation of working_at(a) is considered more precise than ¬working_at(a) (it is easy to verify by comparing the bodies of the rules). It is also shown that two literals L and L′ that are non-contradictory can also disagree if there is a derivation for ¬L. This is the case of illuminated(a) and dark(a) because ¬dark(a). In this case, illuminated(a) is considered more precise than dark(a). However, illuminated(b) is not warranted because neither light_on(b) nor ¬light_on(b) are warranted (since they are contradictory conclusions). Therefore, the DeLP-answers for illuminated(a), working_at(a), ¬dark(a), and dark(b) are true*, the DeLP-answers for ¬working_at(a), dark(a), and ¬dark(b) are* false*, and the DeLP-answer for illuminated(b), light_on(b), and ¬light_on(b) are* undecided*.*

To compare DeLP with our proposal, we adapt the above example by considering that every rule is a defeasible rule, i.e. $P = \Pi \cup \Delta$. We then show separately the results for rooms a and b:

- *The results of a consistent way of reasoning for room a are quite similar to those of DeLP. The literals illuminated(a) and working_at(a) are non-contradictory$_P^{w,w}$, whereas ¬working_at(a) is false$_P^{s,w}$. This happens because working_at(a) has two arguments supporting it, whereas ¬working_at(a) has only one. However, we detect contradiction between dark(a) and ¬dark(a), and so both are undefined$_P^{w,w}$.*

  In a paraconsistent way of reasoning: illuminated(a) is non-contradictory$_P^{s,w}$, whereas ¬dark(a), dark(a), ¬working_at(a) and working_at(a) are contradictory$_P^{s,w}$. Therefore, since we have no comparison criterion as DeLP to choose between contradictory arguments, we conclude both as undefined$_P^{w,w}$ or contradictory$_P^{s,w}$.

- *In a consistent way of reasoning, dark(b) is non-contradictory$_P^{w,w}$, ¬dark(b) is false$_P^{w,w}$, whereas illuminated(b), light_on(b), ¬light_on(b) are undefined$_P^{w,w}$. Those results are similar to DeLP.*

  However, we present different results by applying a paraconsistent way of reasoning: dark(b) is non-contradictory$_P^{s,w}$. We detect contradiction between light_on(b) and ¬light_on(b) and so both are contradictory$_P^{s,w}$. Consequently,

illuminated(b) is based-on-contradiction$_P^{s,w}$. Note that such results may "signal" that something is wrong with room b. Therefore, in this case of contradiction we argue that we present results more intuitive than DeLP.

5.3 Argument-based Negotiation

Negotiation has its origin in both Distributed Problem Solving (DPS), where the agents are assumed to be cooperative, and Multi-agent Systems (MAS), where the agents are supposed to be moved by self-interest. However, there are also proposals in MAS of mechanisms for cooperative agents who need to resolve conflicts that arise from conflicting beliefs about different aspects of their environment. In DPS, negotiation is used for distributed planning and distributed search for possible solutions for hard problems. In MAS, an abstract negotiation framework can be viewed in terms of its negotiating agents (with their internal motivations, decision mechanisms, knowledge-bases, etc.) and the environment in which these agents interact.

The multi-agent paradigm offers a powerful set of metaphors, concepts, and techniques for conceptualizing, designing, implementing, and verifying complex distributed systems. An agent is viewed in [RRJ+04] as an encapsulated computer system that is situated in an environment and is capable of flexible, autonomous action in order to meet its design objectives. Most often, such agents need to interact in order to fulfill their objectives or improve their performance. Generally speaking, different types of interaction mechanisms suit different types of environments and applications. Thus, agents may need mechanisms that facilitate information exchange, coordination (in which the agents arrange their individual activities in a coherent manner), collaboration (in which agents work together to achieve a common objective), and so on. One such type of interaction that is gaining increasing prominence in the agent community is *automated negotiation*[8].

Several interaction and decision mechanisms for negotiation in a multi-agent setting have been proposed and discussed. The three major classes of approaches applied to multi-agent settings are game-theoretic analysis, heuristic approaches, and argumentation-based approaches. A brief review and a comparison of these three approaches is presented in [RRJ+04]. Those approaches consider the mechanism (or protocol), the agent strategies within the rules of the protocol, and the outcome (i.e. a deal or a conflict over a negotiation set). The latter is the result of the mechanism and the participant strategies applied in the negotiation process.

In most game-theoretic and heuristic models, agents exchange *proposals* (i.e. potential agreements or potential deals). Agents are not allowed to exchange any

[8] In the following we write *negotiation* for *automated negotiation*.

additional information other than what is expressed in the proposal itself. Another limitation of conventional approaches to negotiation is that agent's utilities or preferences are usually assumed to be completely characterized prior to the interaction. Thus, an agent is assumed to have a mechanism by which it can assess and compare any two proposals. In more complex negotiation situations, such as trade union negotiations, agents may well have incomplete information which limits this capability. Then, the agents might have inconsistent or uncertain beliefs about the environment, have incoherent preferences, have unformed or undetermined preferences, and so on. To overcome these limitations, the processes of acquiring information, resolving uncertainties, or revising preferences often take place as part of the negotiation process itself. A further drawback of traditional models for negotiation is that agent's preferences over proposals are often assumed to be proper (in the sense that they reflect the true benefit the agent receives from satisfying these preferences). Finally, game-theoretic and heuristic approaches assume that agent's utilities or preferences are fixed. One agent cannot directly influence another agent's preference model, or any of its internal mental attitudes (e.g., beliefs, desires, goals, etc.) that generate its preference model. A rational agent would only modify its preferences upon receipt of new information.

We can conclude that the *Game-theoretic approach* and the *Heuristic-based approach* share some limitations such as (1) agents exchange proposals (potential agreements or deals) but they are not allowed to exchange any additional information to justify such proposals; (2) agent's utilities or preferences are assumed to be completely characterized prior to the interaction; (3) agents' preferences are assumed to be proper (i.e. they reflect the truth benefit the agents gets); (4) both approaches assume that the agents' utilities or preferences are fixed (i.e. agents cannot influence on other agents' preference models or internal mental attitudes). Argumentation-based approaches to negotiation attempt to overcome the above limitations by allowing agents to exchange additional information, or to *argue* about their beliefs and other mental attitudes, during the negotiation process.

An argument-based negotiation protocol [PSJ98] is basically based on the exchange of proposals, critiques, counter-proposals, and explanations. It usually proceeds in a series of rounds, with every agent making a proposal - a kind of solution to the problem - at each round. One agent generates a proposal and other agents review it. If some agent does not like the proposal, it rejects the proposal, and generates a kind of feedback, either as a counter-proposal (i.e. an alternative proposal generated in response to the initial proposal) or as a critique (i.e. comments on which parts of the proposal the agents likes or dislikes). Then, every agent, including the agent that generated the first proposal, reviews the feedback. Based on the reviewing of the feedback, the proponent may generate a proposal to lead to an agreement. In addition of generating proposals, counter-proposals,

and critiques, the agents can make the proposal more attractive by providing additional meta-level information in the form of an argument. The process is then repeated. It is assumed that a proposal becomes a solution when it is accepted by all agents. Furthermore, argumentation may be used both at the level of an agent's internal reasoning and at the level of negotiation between the agents, and where preferences can change. Therefore, argumentation can be seen as a more sophisticated exchange of information in a negotiation protocol, or as a model for reasoning based on the construction and comparison of arguments.

According to [RRJ+04], an argumentation-based negotiation is viewed as a form of interaction in which a group of agents, with conflicting interests and a desire to cooperate, try to come to a mutually acceptable agreement on the division of scarce resources, not all of which can be simultaneously satisfied. A resource is understood as commodities, services, time, money, etc. In short, anything that is required to achieve something. According to *Argumentation in Multi-Agent Systems* (ArgMAS 2010), argumentation can be abstractly defined as the formal interaction of different arguments for and against some conclusion (eg., a proposition, an action intention, a preference, etc.). An agent may use argumentation techniques to perform individual reasoning, in order to resolve conflicting evidence or to decide between conflicting goals. Multiple agents may also use dialectical argumentation in order to identify and reconcile differences between themselves, through interactions such as negotiation, persuasion, and joint deliberation. In *Computational Models for Argumentation in Multiagent Systems* (2005), a multiagent system consists of a number of agents, which interact with one-another. In the most general case, agents will be acting on behalf of users with different goals and motivations. To successfully interact, they will require the ability to *cooperate*, *coordinate*, and *negotiate* with each other, much as people do. We can conclude that argumentation provides tools for designing, implementing, and analyzing sophisticated forms of interaction in multi-agent systems. Moreover, a single agent may use argumentation techniques to perform its individual reasoning because it needs to make decisions under complex preferences policies, in a highly dynamic environment. Therefore, argumentation has made solid contributions to the practice of negotiation in multi-agent systems.

According to [AC02], the argumentation encopasses two views of arguments: (i) a local view that intends to give support in favor or against a conclusion, and (ii) a global view that intends to define acceptable arguments. We follow the second view, and so we focus our attention on that. Formal argumentation systems (e.g. [SL92b, Vre97, Pol01, Dun93, Dun95, PS97]) are characterized by representing precisely some of these features of argumentation via formal languages, and by applying formal inferences techniques. The different approaches, which have been developed for reasoning within an argumentation system, use one of the following

kinds of acceptability: (i) individual acceptability [EGH95] where an acceptability level is assigned to a given argument on the basis of the existence of direct defeaters; or (ii) joint acceptability [Dun93, Dun95] where the set of the accepted arguments must defend itself against any defeater. These two notions of acceptability have been most often defined purely on the basis of defeaters. The resulting evaluation of arguments is only based on the interactions between (direct or indirect) defeaters. However, other criteria may be taken into account for comparing arguments such as for instance, preference [AC02], specifity [SL92b], or explicit priorities [PS97]. Below we give a brief overview of proposals that define acceptable arguments, and draw some comparisons with our work.

The preference-based argumentation approach shows that it is not realistic to assume that arguments have all the same strength (e.g. [Dun95]'s abstract argumentation framework) since it may be the case that an argument relies on certain information, while another argument is built from less certain ones. The former argument has to be stronger than the latter. Therefore, preferences have been introduced into argumentation theory to solve conflicts where a preference relation captures differences in arguments' strengths. An extension of Dung's framework [AC02] is proposed as a preference-based argumentation approach. The idea behind this extension is that an attack from an argument a to an argument b fails if b is stronger than a. To do so, it takes as input a set $\mathcal{A}rg$ of arguments, an attack relation $\mathcal{R}$, and a (partial or total) preorder[9] $\geq$. This preorder is a preference relation between arguments. The expression $(a, b) \in \geq$ or $a \geq b$ means that the argument a is at least as strong as b. The symbol $>$ denotes the strict relation associated with $\geq$. Indeed, $a > b$ iff $a \geq b$ and *not* $(b \geq a)$. From the two relations $\mathcal{R}$ and $\geq$, a new binary relation, Def, is defined as follows: $a\ Def\ b$ iff $a\mathcal{R}b$ and *not* $(b > a)$. This means that among all the attacks in $\mathcal{R}$, only the ones that hold between incomparable and indifferent arguments and the ones that agree with the preference relation are kept. In order to evaluate the acceptability of arguments, Dung's acceptability semantics are applied to the framework $< \mathcal{A}rg, Def >$. [AV09] shows that this proposal gives unintended results with the following example:

Example 42 *Let us to consider the case of an agent who wants to buy a given violin. An expert says that the violin in question is produced by Stradivari (s), that's why it is expensive(e ← s). This agent has thus an argument a_1 whose conclusion is "the violin is expensive". Suppose now that the 3-years old son of this agent says that the violin was not produced by Stradivari (¬s). Thus, an argument a_2 which attacks a_1 is given. In sum, $\mathcal{A}rg = \{a_1, a_2\}$ and $\mathcal{R} = \{(a_2, a_1)\}$. According to Dung's framework, argument a_2 wins. This is inadmissible, especially since it*

[9] A binary relation is a preorder iff it is reflexive and transitive.

is clear that an argument of an expert is stronger than an argument given by a 3-years old child. In the framework presented in [AC02], the fact that a_1 is stronger than a_2 is taken into account. Thus, the relation $\geq = \{(a_1, a_1), (a_2, a_2), (a_1, a_2)\}$ is available. However, in this framework the relation Def is empty. Consequently, the arguments a_1 and a_2 are in the unique preferred extension[10]. This means that this extension is not conflict-free[11]. Moreover, both s and $\neg s$ are deduced.

We can further evaluate the above example by assuming an argument a_3 for *stradivari*. We then obtain a new $\mathcal{A}rg = \{a_1, a_2, a_3\}$ such that $\mathcal{R} = \{(a_2, a_1), (a_2, a_3), (a_3, a_2)\}$. According to Dung's framework, no argument wins. This is an inadmissible result because the argument a_2 (from a child) invalidates both arguments a_1 and a_3 (from an expert). That is also our result if we apply a (centralized) consistent way of reasoning, $F_P^{w,w}$, over an ELP program $P = \{s; \neg s; e \leftarrow s\}$. If we apply a (centralized) paraconsistent way of reasoning, $F_P^{s,w}$, all arguments are acceptable which is also an inadmissible result because it assigns all those arguments the same "strength". However, we may represent such a problem as a MAS:

$$\begin{aligned}\mathcal{A} = \{ \quad & < expert, \{s;\ e \leftarrow s\}, \{expert\}, \{expert, child\} >, \\ & < child, \{\neg s\}, \{child, expert\}, \{child, expert\} > \ \}\end{aligned}$$

we define a preference between expert and child, i.e. the child must argue with the expert, but not vice-versa. Moreover, we model that both agents cooperate with each other and so they deduce the same truth value for every literal in $\mathcal{H}(\mathcal{A})$. In a paraconsistent way of reasoning, e, s and $\neg s$ are $\text{true}_{Ag}^{s,w}$ $(Ag = \{child, expert\})$ which are inadmissible results. However, in a consistent way of reasoning, e and s are $\text{true}_{Ag}^{w,w}$, and $\neg s$ is $\text{false}_{Ag}^{w,w}$ $(Ag = \{child, expert\})$. Therefore, we solve the problem presented in [AV09]'s example by modelling cooperative and argumentative agents to obtain preference over rules from an expert.

[AV09] proposes a new preference-relation argumentation framework that ensures being conflict-free w.r.t. the attack relation and so solves the problem presented in Example 42. The proposal has similar results to the ones to those presented above. Thus, this framework recovers Dung's acceptability extensions, viz. preferred extensions and grounded extension. Since we follow the grounded extension, we will focus our attention on that. The framework takes as input three elements: a set of arguments $\mathcal{A}rg$, an attack relation $\mathcal{R}$, and a (partial or total) preorder $\geq$. It returns extensions that are subsets $\mathcal{E}$ of $\mathcal{A}rg$. These extensions

[10] A set of arguments is a preferred extension iff it is a maximal (w.r.t. $\subseteq$) admissible set. A set of arguments is admissible iff it defends all its elements. A set of arguments defends an argument a iff $\forall b \in \mathcal{A}rg$ if $b\mathcal{R}a$, then $\exists c \in \mathcal{B}$ such that $c\mathcal{R}b$ whether the framework $< \mathcal{A}rg, Def >$. For details see Section 2.2

[11] A set $\mathcal{B}$ of arguments is conflict-free iff $\neg\exists a, b \notin \mathcal{B}$ such that $a\mathcal{R}b$.

satisfy the two following basic requirements, viz. conflict-freedom and generalization. The former ensures safe results in the sense that inconsistent conclusions in $\mathcal{E}$ are avoided. The latter captures the idea that an attack fails in case the attacker is weaker than its target. Moreover, it states that the proposed approach extends Dung's framework, i.e. it refines its acceptability semantics. Preferences relations, denoted by $\succeq$, between the different conflict-free sets $\mathcal{E}$ of arguments are defined: $\mathcal{E} \succ \mathcal{E}'$ iff $\mathcal{E} \succeq \mathcal{E}'$ and *not* $(\mathcal{E}' \succeq \mathcal{E})$. Then *maximal elements* and the new *preference-based argumentation* (PAF) are defined as follows:

Definition 70 (Maximal Elements) *Let $\mathcal{E}$ be a conflict-free set of arguments. $\mathcal{E}$ is* maximal *w.r.t. $\succeq$ iff:*

1. $(\forall \mathcal{E}' \subseteq \mathcal{A}rg)$ $((\mathcal{E}$ *is conflict-free*$) \Rightarrow (\mathcal{E} \succeq \mathcal{E}'))$
2. *No strict superset of $\mathcal{E}$ is conflict-free and verifies (1)*

Let $\succeq_{max}$ denote the set of maximal sets w.r.t. $\succeq$.

Definition 71 (PAF) *A* PAF *is a tuple $(\mathcal{A}rg, \mathcal{R}, \geq)$, where $\mathcal{A}rg$ is a set of arguments, $\mathcal{R}$ is an attack relation, and $\geq$ is a (partial or total) preorder on $\mathcal{A}rg$. Extensions of $(\mathcal{A}rg, \mathcal{R}, \geq)$ are the maximal elements of a relation $\succeq \subseteq 2^{\mathcal{A}rg} \times 2^{\mathcal{A}rg}$ that satisfies the two basic requirements.*

Then, a relation which generalizes grounded semantics [Dun95] is defined. The basic idea behind this relation is that a set is not worse than another if it can strongly defend all its arguments against all attacks that come from the other set. The notion of *strong defense* is generalized by taking into account preference between arguments: an argument has either to be preferred to its attacker or has to be defended by arguments that themselves can be strongly defended without using the argument in question.

Definition 72 (Strong Defense) *Let $\mathcal{E}' \subseteq \mathcal{A}rg$. $\mathcal{E}'$* strongly defends *an argument x from attacks of a set $\mathcal{E}$, denoted by $sd(x, \mathcal{E}', \mathcal{E})$ iff $(\forall y \in \mathcal{E})$ if $(((y, x) \in \mathcal{R} \wedge (x, y) \notin >)$ or $((x, y) \in \mathcal{R} \wedge (y, x) \in >))$ then $((\exists z \in \mathcal{E}' \backslash \{x\})$ such that $((z, y) \in \mathcal{R} \wedge (y, z) \notin > \wedge sd(z, \mathcal{E}' \backslash \{x\}, \mathcal{E})))$. If the third argument of sd is not specified, then $sd(x, \mathcal{E}) \equiv sd(x, \mathcal{E}, \mathcal{A}rg)$.*

Example 43 *This example is taken from [AV09]. Let $\mathcal{A}rg = \{a, b, c\}$, $\geq = \{(a, a), (b, b), (a, b)\}$ and $\mathcal{R} = \{(a, b), (b, a), (b, c), (c, b)\}$. The conflict-free sets of arguments are: $\mathcal{E}_1 = \emptyset$, $\mathcal{E}_2 = \{a\}$, $\mathcal{E}_3 = \{b\}$, $\mathcal{E}_4 = \{c\}$, and $\mathcal{E}_5 = \{a, c\}$. The relations are: $\mathcal{E}_2 \succeq \mathcal{E}_1$, $\mathcal{E}_3 \succeq \mathcal{E}_1$, $\mathcal{E}_4 \succeq \mathcal{E}_1$, $\mathcal{E}_5 \succeq \mathcal{E}_1$, $\mathcal{E}_5 \succeq \mathcal{E}_4$, $\mathcal{E}_5 \succeq \mathcal{E}_2$, $\mathcal{E}_5 \succeq \mathcal{E}_3$, $\mathcal{E}_4 \succeq \mathcal{E}_3$, $\mathcal{E}_3 \succeq \mathcal{E}_4$, and $\mathcal{E}_2 \succeq \mathcal{E}_3$. Therefore, $\succeq_{max} = \mathcal{E}_\triangledown$. It holds that $sd(a, \{a\}, \{b\})$ since a is strictly prefered to b thus it can defend itself. However, $\neg sd(b, \{b\}, \{c\})$ holds*

because b cannot defend itself against c. On the other hand, it does hold that $sd(c, \{a, c\}, \{b\})$ *since a can defend c against b and a is protected from b since it is strictly prefered to it. Therefore, a and c are acceptable arguments, whereas b is not.*

To compare PAF with our proposal, we need to adapt the proposed argumentation set of arguments $\mathcal{A}rg$ *to an MAS. First, we assume that the above argument a (resp. b, c) is* $A_a = [a \leftarrow not\ b]$ *(resp.* $A_b = [b \leftarrow not\ a, not\ c]$, $A_c = [c \leftarrow not\ b]$*). Then, we specify a multi-agent set* $\mathcal{A} = \{Ag_1, Ag_2\}$ *such that*

$$\begin{array}{ll} Ag_1 = & < 1, \{a \leftarrow not\ b\}, \{1\}, \{1, 2\} > \\ Ag_2 = & < 2, \{b \leftarrow not\ a, not\ c;\ c \leftarrow not\ b\}, \{1, 2\}, \{1, 2\} > \end{array}$$

This example has no explicit negation which means that there are no contradictory literals, and so both consistent and paraconsistent ways of reasoning have the same result. In the following we show, step by step, how we obtain the $lfp(F^{s,s}_{\mathcal{A}})$*, i.e. the set of acceptable arguments. First of all, we determine the set of strong local arguments of* $\mathcal{A}$*:*

$$\begin{array}{rl} LA^s(\mathcal{A}) = \{ & (1, []),\ (1, [a \leftarrow not\ b]), \\ & (2, []),\ (2, [c \leftarrow not\ b]) \\ & (2, [b \leftarrow not\ a, not\ c])\ \ \} \end{array}$$

and the set of available arguments of $\mathcal{A}$ *given* $LA^s(\mathcal{A})$ *is*

$$\mathcal{A}rgs(\mathcal{A}) = \ LA^s(\mathcal{A})\ \cup \{\ \ (2, [a \leftarrow not\ b])\ \ \}$$

- *Let* $S^0 = \emptyset$*. Since* $Atts^0 = \emptyset$*, the set of opposing arguments is*

$$Def^0 = Def_{s,s}(S^0) = \mathcal{A}rgs(\mathcal{A})$$

The set of proposed arguments is $LA^s(\mathcal{A})$*, resulting from* $Av(S^0)$*. Then we determine the following attacks*

opposing argument	*proposed argument*
	$(1, [a \leftarrow not\ b])$
$(1, [a \leftarrow not\ b])$	$(2, [b \leftarrow not\ a, not\ c])$
$(2, [a \leftarrow not\ b])$	$(2, [b \leftarrow not\ a, not\ c])$
$(2, [b \leftarrow not\ a, not\ c])$	$(2, [c \leftarrow not\ b])$

Therefore,

$$S^1 = F^{s,s}_{\mathcal{A}}(S^0) = \{\ \ (1, [a \leftarrow not\ b]), (1, []), (2, [])\ \ \}$$

Remark that $(2, [b \leftarrow not\ \ a, not\ \ c])$ *is attacked by arguments for* a *of both agents, and that* $(2, [a \leftarrow not\ \ b])$ *is built by cooperation with* 1. *Furthermore,* $(1, [a \leftarrow not\ \ b])$ *is not attacked because agent* 2 *does not argue with agent* 1. *Such situation permits that every argument of agent* 1 *has preference over arguments of agent* 2.

- *Since* $Atts^1 = \{(1, [a \leftarrow not\ \ b]), (2, [a \leftarrow not\ \ b]), (2, [b \leftarrow not\ \ a, not\ \ c]), (2 \leftarrow not\ \ b)\}$, *the argument* $(2, [b \leftarrow not\ \ a, not\ \ c])$ *in* $\mathcal{A}rgs(\mathcal{A})$ *is rejected and so*

$$Def^1 = Def_{s,s}(S^1) = \{\ \ (1, [a \leftarrow not\ \ b]), (2, [a \leftarrow not\ \ b])\ \ \}$$

The proposed arguments are obtained from $Av(S^1)$, *the attacks are*

opposing argument	*proposed argument*
	$(1, [a \leftarrow not\ \ b])$
	$(2, [a \leftarrow not\ \ b])$
$(1, [a \leftarrow not\ \ b])$	$(2, [b \leftarrow not\ \ a, not\ \ c])$
$(2, [a \leftarrow not\ \ b])$	$(2, [b \leftarrow not\ \ a, not\ \ c])$
	$(2, [c \leftarrow not\ \ b])$

and so

$$S^2 = F^{s,s}_{\mathcal{A}}(S^1) = \{\ \ (1, [a \leftarrow not\ \ b]), (2, [a \leftarrow not\ \ b]), (2, [c \leftarrow not\ \ b]), (1, []), (2, [])\ \ \}$$

- *Since* $F^{s,s}_{\mathcal{A}}(S^2) = S^2$, *we then obtain the* $lfp(F^{s,s}_{\mathcal{A}})$. *We can further say the justified*$^{s,s}_{\mathcal{A}}$ *arguments are in* S^2, *the overruled*$^{s,s}_{\mathcal{A}}$ *argument is* $(2, [b \leftarrow not\ \ a, not\ \ c])$, *and there is no defensible*$^{s,s}_{\mathcal{A}}$ *argument. Therefore, we obtain results similar to the (centralized) preference-relation argumentation framework from [AV09].*

We show in the above example that we can specify preference rules between two agents by defining that the agent with less priority has to argue with the other one. If both agents cooperate with each other, we garantee that the conclusions will be the same. The preference relation between arguments has to be individually specified in [AV09] whereas we globaly define the preference relation for differents (cooperative and argumentative) agents in the same multi-agent setting. So each agent has some level of preference rules over the others ones.

By defining sets of (cooperative and argumentative) agents, we apply our proposal in different aproaches such as those presented in the Section 4.5. On that, we can model a situation where arguments have to be deduced without a "global view of the world". For instance, assume a multi-agent setting $\mathcal{A} = \{Ag_1, Ag_2, \ldots, Ag_n\}$

and an agent Ag_i that does not argue with agent Ag_j ($1 \leq j \leq n$ and $j \neq i$). Ag_i deduces the truth value of any literal L without consider the fact that the arguments for L might be attacked by arguments in some A_j. [AV09]'s framework needs to define a preference relation over all arguments of both Ag_i and Ag_j to obtain the same result as our. Therefore, we consider our proposal easier to model such situation.

[BC03] also extends [Dun95]'s argumentation framework. It assumes that each argument promotes a value, and a preference between two arguments comes from importance of the respective values that are promoted by two arguments. In [BCA09], an (abstract) *Value-based argumentation framework* is presented as follows:

Definition 73 *A* value-based argumentation framework *(VAF) is a 5-tuple:*

$$VAF =< \mathcal{A}rg, \mathcal{R}, \mathcal{V}, val, P >$$

where $\mathcal{A}rg$ is a finite set of arguments, $\mathcal{R}$ is an irreflexive binary relation on $\mathcal{A}rg$, $\mathcal{V}$ is a non-empty set of values, val is a function which maps elements of $\mathcal{A}$ to elements of $\mathcal{V}$, and P is the set of possible audiences (i.e. total orders on $\mathcal{V}$). An argument a relates to value v if accepting $\mathcal{A}rg$ promotes or defends v: the value in question is given by $val(a)$. For every $a \in \mathcal{A}rg$, $val(a) \in \mathcal{V}$.

When the VAF is considered by a particular audience, the ordering of values is fixed. Then an *Audience Specific VAF* is as following:

Definition 74 *An* audience specific value-based argumentation framework *(AVAF) is a 5-tuple:*

$$VAF_a =< \mathcal{A}rg, \mathcal{R}, \mathcal{V}, val, Valpref_a >$$

where $\mathcal{A}rg$, $\mathcal{R}$, $\mathcal{V}$, and val are as for a VAF, a is an audience in P, and $ValPref_a \subseteq \mathcal{V} \times \mathcal{V}$ is a preference relation (transitive, irreflexive, and asymmetric), reflecting the value preferences of audience a. The $AVAF$ relates to the VAF in that $\mathcal{A}rg$,$\mathcal{R}$,$\mathcal{V}$, and val are identical, and $ValPref_a$ is the set of preferences derivable from the ordering $a \in P$ in the VAF.

The purpose of introducing VAFs is to distinguish between one argument *attacking* another, and that attack *succeeding*, so that the *attacked* argument may or may not be defeated. Whether the attack succeeds depends on the value order of the audience considering the VAF. Then the notion of *defeat* is defined as follows:

Definition 75 *An argument $A \in AVAF$ $defeats_a$ an argument $B \in AVAF$ for an audience a iff both $\mathcal{R}(A, B)$ and not $(val(A), val(B)) \in ValPref_a$.*

Then, various notions of the status of arguments are defined as:

Definition 76 *An argument $A \in \mathcal{A}rg$ is* acceptable-to-audience-a ($acceptable_a$) *w.r.t. a set of arguments S ($acceptable_a(A, S)$) if*

$$(\forall x)((x \in \mathcal{A} \ \wedge \ defeats_a(x, A)) \rightarrow (\exists y)(y \in S \ \wedge \ defeats_a(y, x))).$$

Definition 77 *A set S of arguments is* conflict-free-for-audience-a *if*

$$(\forall x)(\forall y)((x \in S \ \wedge \ y \in S) \rightarrow (\neg\mathcal{R}(x, y) \vee (val(y), val(x)) \in ValPref_a)).$$

Definition 78 *A conflict-free-for-audience-a set of arguments S is* admissible-for-an-audience-a *if*

$$(\forall x)(x \in S \rightarrow acceptable_a(x, S)).$$

Definition 79 *A set of arguments S in a value-based argumentation framework VAF is a* prefered-extension-for-audience-a ($prefered_a$) *if it is a maximal (w.r.t. set of inclusion) admissible-for-audience-a subset of $\mathcal{A}$.*

Example 44 *Let $\mathcal{A} = \{Ag_{blue}, Ag_{red}\}$ such that each agent is*

$$Ag_{blue} = < \ blue, \left\{ \begin{array}{l} a \leftarrow not \ \ d; \ d \leftarrow not \ \ c; \\ g \leftarrow not \ \ f; \ h \leftarrow not \ \ g \end{array} \right\}, \ \{blue\}, \ \{blue\} \ >$$

$$Ag_{red} = < \ red, \left\{ \begin{array}{l} b \leftarrow not \ \ a; \ c \leftarrow not \ \ b; \\ f \leftarrow not \ \ e; \ e \leftarrow not \ \ h \end{array} \right\}, \ \{red, blue\}, \ \{red, blue\} \ >$$

This example is adapted from [BCA09], a VAF with values red and blue. On that, there are two prefered extensions, according to whether $red > blue$, or $blue > red$. If $red > blue$, the preferred extension is $\{e, g, a, b\}$; and if $blue > red$, $\{e, g, d, b\}$. Moreover, e and g are objectively acceptable*, and c, f and h are* indefensible*. If $red > blue$ (resp. $blue > blue$) then a (resp. d) is* subjectively acceptable*, and b (resp. a) is defensible.*

First of all, we determine the set of strong local arguments of $\mathcal{A}$:

$$\begin{array}{rl} LA^s(\mathcal{A}) = \{ & (blue, []), \ (blue, [a \leftarrow not \ \ d]), \ (blue, [d \leftarrow not \ \ c]), \\ & (blue, [g \leftarrow not \ \ f]), \ (blue, [h \leftarrow not \ \ g]), \\ & (red, []), \ (red, [b \leftarrow not \ \ a]), \ (red, [c \leftarrow not \ \ d]), \\ & (red, [f \leftarrow not \ \ e]), \ (red, [e \leftarrow not \ \ h]) \ \} \end{array}$$

and the set of available arguments of $\mathcal{A}$ given $LA^s(\mathcal{A})$ is

$$\begin{array}{rl} \mathcal{A}rgs(\mathcal{A}) = & LA^s(\mathcal{A}) \ \cup \{ \ (red, [a \leftarrow not \ \ d]), (red, [d \leftarrow not \ \ c]), \\ & (red, [g \leftarrow not \ \ f]), (red, [h \leftarrow not \ \ g]) \ \} \end{array}$$

- *Let* $S^0 = \emptyset$. *Since* $Atts^0 = \emptyset$, *the set of opposing arguments is*

$$Def^0 = Def_{s,s}(S^0) = \mathcal{A}rgs(\mathcal{A})$$

The set of proposed arguments is $LA^s(\mathcal{A})$, *resulting from* $Av(S^0)$. *Then we determine the following attacks*

opposing argument	*proposed argument*
$(blue, [d \leftarrow not\ \ c])$	$(blue, [a \leftarrow not\ \ d])$
$(blue, [d \leftarrow not\ \ c])$	$(red, [a \leftarrow not\ \ d])$
$(blue, [a \leftarrow not\ \ d])$	$(red, [b \leftarrow not\ \ a])$
$(red, [a \leftarrow not\ \ d])$	$(red, [b \leftarrow not\ \ a])$
$(red, [b \leftarrow not\ \ a])$	$(red, [c \leftarrow not\ \ b])$
	$(blue, [d \leftarrow not\ \ c])$
$(red, [c \leftarrow not\ \ b])$	$(red, [d \leftarrow not\ \ c])$
$(blue, [h \leftarrow not\ \ g])$	$(red, [e \leftarrow not\ \ h])$
$(red, [h \leftarrow not\ \ g])$	$(red, [e \leftarrow not\ \ h])$
$(red, [e \leftarrow not\ \ h])$	$(red, [f \leftarrow not\ \ e])$
	$(blue, [g \leftarrow not\ \ f])$
$(red, [f \leftarrow not\ \ e])$	$(red, [g \leftarrow not\ \ f])$
$(blue, [g \leftarrow not\ \ f])$	$(blue, [h \leftarrow not\ \ g])$
$(blue, [g \leftarrow not\ \ f])$	$(red, [h \leftarrow not\ \ g])$

and so

$$S^1 = F^{s,s}_{\mathcal{A}}(S^0) = \{\ \ (blue, [d \leftarrow not\ \ c]), (blue, [g \leftarrow not\ \ f]), (red, []), (blue, [])\ \ \}$$

Remark that some arguments of red *are attacked by both arguments of* blue *and arguments of* red *(in this case, built by cooperation with* blue, *e.g.* $(red, [a \leftarrow not\ \ d])$*). However, arguments of* blue *are only attacked by arguments of* blue. *Such situation permits that every argument of* blue *has preference over arguments of* red.

- *Since* $Atts^1 = S^1 - \{(red, []), (blue, [])\}$, *the arguments* $(blue, [a \leftarrow not\ \ d])$, $(red, [a \leftarrow not\ \ d])$, $(blue, [h \leftarrow not\ \ g])$, $(red, [h \leftarrow not\ \ g])$ *in* $\mathcal{A}rgs(\mathcal{A})$ *are rejected and so*

$$Def^1 = Def_{s,s}(S^1) = Def^0 - \{\ \ (blue, [a \leftarrow not\ \ d], (red, [a \leftarrow not\ \ d], (blue, [h \leftarrow not\ \ g], (red, [h \leftarrow not\ \ g]\ \ \}$$

Then proposed arguments are obtained from $Av(S^1)$*, and the attacks are*

opposing argument	*proposed argument*
	$(red, [b \leftarrow not\ \ a])$
$(red, [b \leftarrow not\ \ a])$	$(red, [c \leftarrow not\ \ b])$
	$(blue, [d \leftarrow not\ \ c])$
$(red, [c \leftarrow not\ \ b])$	$(red, [d \leftarrow not\ \ c])$
	$(red, [e \leftarrow not\ \ h])$
$(red, [e \leftarrow not\ \ h])$	$(red, [f \leftarrow not\ \ e])$
	$(blue, [g \leftarrow not\ \ f])$
$(red, [f \leftarrow not\ \ e])$	$(red, [g \leftarrow not\ \ f])$

and so

$$S^2 = F^{s,s}_{\mathcal{A}}(S^1) = \{\ \ (red, [b \leftarrow not\ \ a]), (blue, [d \leftarrow not\ \ c]), (red, [e \leftarrow not\ \ h]), (blue, [g \leftarrow not\ \ f]), (red, []), (blue, [])\ \ \}$$

- *Since* $Atts^2 = S^2 - \{(red, []), (blue, [])\}$*, the arguments* $(red, [c \leftarrow not\ \ b])$*,* $(red, [f \leftarrow not\ \ e])$ *in* $\mathcal{A}rgs(\mathcal{A})$ *are rejected and so*

$$Def^2 = Def_{s,s}(S^2) = Def^1 - \{\ \ (blue, [d \leftarrow not\ \ c]), (red, [e \leftarrow not\ \ h]), (blue, [g \leftarrow not\ \ f])\ \ \}$$

Then proposed arguments are obtained from $Av(S^1)$*. Since no attack is determined for the proposed arguments, then*

$$S^3 = F^{s,s}_{\mathcal{A}}(S^2) = \{\ \ (red, [b \leftarrow not\ \ a]), (blue, [d \leftarrow not\ \ c]), (red, [d \leftarrow not\ \ c]), (red, [e \leftarrow not\ \ h]), (blue, [g \leftarrow not\ \ f]), (red, [g \leftarrow not\ \ f]), (red, []), (blue, [])\ \ \}$$

- *Finally,* $F^{s,s}_{\mathcal{A}}(S^3) = S^3$*. We can say that the acceptable arguments of* $\mathcal{A}$ *are in* S^3*, the* $lfp(F^{s,s}_{\mathcal{A}})$*.*

Moreover, the justified$^{s,s}_{\mathcal{A}}$ *arguments are in* S^3*, the overruled*$^{s,s}_{\mathcal{A}}$ *are in the set* $\{(blue, [a \leftarrow not\ \ d]), (red, [a \leftarrow not\ \ d]), (blue, [h \leftarrow not\ \ g]), (red, [h \leftarrow not\ \ g]), (red, [c \leftarrow not\ \ b]), (red, [f \leftarrow not\ \ e])\}$*, and there is no defensible*$^{s,s}_{\mathcal{A}}$ *argument.*

Therefore, we obtain similar results as the (centralized) Bench-Capon proposal [BC03]. Note that we cannot determine which argument is subjectively/objectively acceptable because we follows the unique-status approach. However, we could deduce that both arguments for d *and* g *of blue are acceptable given the preference relation between both agents.*

We show in the above example that we specify preference rules between two agents by defining that both agents cooperate with each other, and only the agent with less priority has to argue with the other agent. Since both agents cooperate with each other, the truth value of any literal will be the same. Similar to the previous approach, [BC03]'s approach cannot be applied for obtainning a conclusion in an agent without considering the knowlegde of other agents.

5.4 Some conclusions

We relate our work with proposals in semantics of abstract argumentation systems, defeasible reasoning, and argumentation-based negotiation. On that, we show that

- the ideal semantics[DMT06] allows the acceptance of a set of arguments possibly larger that our (distributed) argumentation-based negotiation when every agent cooperates with all agents in $\mathcal{A}$. However, we obtain different results for any other configuration of sets of argumentative and cooperative agents;
- we do not deal with 'indirect attack', and so the centralized prudent semantics [CMDM05] is more restrictive than our characteristic function;
- we generalized the [PS97]'s proposal;
- our results are more intuitive than DeLP [Gar00] when in presence of contradiction; and
- we are able to deal with preferences as in the preference-based argumentation framework [AV09], and the value-based argumentation framework [BC03]. These frameworks define a preference relation between arguments, and we show that we can model prefence relation between set of rules ranging from consistent to a paraconsistent way of reasoning.

Therefore, our proposal is general enough to capture some of these approaches, without losing the opportunity of having some other results when

- two agents (1) neither argue nor cooperate with each other, or (2) both argue with each other. In this case there is no preference between agents' rules and so every argument has the same "strength";
- an agent Ag_1 has to argue with agent Ag_2, but not vice-versa. In this case the rules of Ag_2 have more "strength" than Ag_1, and so we model preference relation between those sets of rules; and

- two agents do not cooperate with each other, and so the truth value of a literal L is different, depending on who L is infered.

All of these cases can be applied in the same multi-agent setting with different (cooperative and argumentative) sets of agents. Moreover, since our argumentation is parameterized by the kind of interaction between arguments, we obtain results ranging from a consistent way of reasoning to a paraconsistent way of reasoning; the former is more sceptical than the latter. We assume, in a consistent way, that every contradictory literal has to be undefined. However, in a paraconsistent way, we deal with contradiction and a literal may be true and contradictory, based on contradiction, or non contradictory (for details see Def. 49). None of the above aproaches present such a distinction when in presence of contradiction.

Chapter 6

Conclusions and Future Work

This chapter goes back to the objectives drawn in the introduction, synthesizing the way how the work which unfolded throughout this dissertation has fulfilled them. Then, it outlines some future research aspects that emerged from the work presented herein

---

The main goal of this dissertation was to define an argumentation-based negotiation for agent's knowledge bases. The agent's beliefs are characterized by the relations between its "internal" arguments supporting its beliefs and the "external" arguments supporting the contradictory beliefs of other agents. So in a certain sense, argumentative reasoning is based on the "external stability" of acceptable arguments in a multi-agent setting. It was also a goal to have an argumentation-based semantics that can deduce the acceptability of agent's arguments in a paraconsistent way of reasoning. Moreover, in some applications it may be interesting to easily change from a paraconsistent to a consistent way of reasoning (or vice-versa). Furthermore, the Argumentation-based semantics should be as simple as possible because its definitions will be used in the Argumentation-based Negotiation semantics. Therefore, we first defined a Self-argumentation framework, upon which most of the definitions of the Argument-based Negotiations semantics were constructed. Consequently, we first present the contributions from the Self-argumentation framework, followed by the Argumentation-based Negotiation framework.

Self-Argumentation Framework Our self-argumentation semantics is based on the argumentation metaphor, in the line of the work developed in [Dun95,

PS97, BDKT97, SS02b] for defining semantics of single extended logic programs. In these argumentation-based semantics, rules of a logic program are viewed as encoding arguments of an agent. More precisely, an argument for an objective literal L is a sequence of rules that "proves" L, if all default literals (of the form *not* L') in the body of those rules are assumed true. In other words, arguments encoded by a program can attack — by undercut — each other. Moreover, an argument for L attacks — by rebut — another argument if this other argument assumes its explicit negation (of the form $\neg L$). The meaning of the program is then determined by those arguments that somehow (depending on the specific semantics) can defend themselves from the attacks of other arguments.

We generalize [PS97]'s definition of argument by proposing two kinds of arguments, viz. strong arguments and weak arguments. Having two kinds of arguments, attacks by undercut do not need to be considered. Simply note that rebut is undercut against weak arguments. Therefore, rebut is not considered in our proposal since, as already shown in [SdAMA97, dAMA98b, SS02b], it can be reduced to undercut by considering weaker versions of arguments. We further extend [PS97]'s argumentation-based semantics for *extended logic programs* [PA92] to deal with denials.

Similarly to [Dun95, PS97] we formalize the concept of acceptable arguments with a fixpoint operator. However, the acceptability of an argument may have different results and it depends on which kind of interaction between (strong and weak) arguments is chosen. Therefore, our argumentation semantics assigns different levels of acceptability to an argument and so it can be justified, overruled, or defensible. Moreover, a justified argument can be contradictory, based on contradiction, or non contradictory. Consequently, the truth value of an literal can be true (and either contradictory, based on contradiction, or non contradictory), false, or undefined.

Since our argumentation semantics is parametrised by the kind of interaction between arguments, we obtain results ranging from a consistent way of reasoning to a paraconsistent way of reasoning. In the presence of rules for both literals L and $\neg L$: a consistent way of reasoning neither concludes L nor $\neg L$ as true (even if one of these is a fact); a paraconsistent way of reasoning can conclude that L is true even if it also concludes that $\neg L$ is true. Given that we consider denials in the agent's knowledge base, a consistent way of reasoning does not conclude a given L as true if L is related with the presence of *falsity*; a paraconsistent way of reasoning may conclude L even if it is related with *falsity*.

[CS05] states that "some researchers say that the difference between the

two approaches can be compared with the *skeptical* vs *credulous* attitude towards drawing defeasible conclusions. The multi-status assignment (MSA) is more convenient for identifying sets of arguments that are compatible with each other, and an argument is *genuinely justified* if and only if it is justified in all possible assignments. The unique-status assignment (USA) considers arguments on an individual basis, and so undecided conflicts get the status *not justified* (i.e. overruled or defensible)". Our proposal is an USA approach. However, since our argumentation semantics is parametrised by the kind of interaction between arguments, we obtain results ranging from a consistent way of reasoning to a paraconsistent way of reasoning and, as shown above, the latter is more credulous than the former.

The results obtained through the characteristic function $p\ o$ of a logic program P (cf. Def. 43) coincide with well-founded semantics:

- $WFSX_p$ semantics [ADP95] and $F_P^{s,w}$ (cf. Theorem 33),
- *Grounded extension* [Dun95] and $F_P^{w,w}$ (cf. Theorem 36),
- $WFSX$ [PA92] and $F_P^{w,s}$ (cf. Theorem 34),
- WFS semantics [Prz90] and $F_P^{s,s}$ (cf. Theorem 37).

Argumentation-based Negotiation Framework As already said, we extend the Self-argumentation semantics to an Argumentation-based Negotiation semantics. The goal of this semantics is to define a framework such that agents negotiate by exchanging parts of their knowledge (i.e. arguments) and obtain a consensus concerning their beliefs. In other words, the agents evaluate arguments to obtain a consensus about a common knowledge by either proposing arguments, or trying to built opposing arguments against them. As in the centralized version, the proposed framework is able to deal with mutually inconsistent, and even inconsistent, knowledge bases. Moreover, when in presence of contradiction, it obtains ways of multi-agent setting reasoning, ranging from consistent (in which inconsistencies lead to no result) to paraconsistent.

The argumentation-based negotiation deals with incomplete knowledge (i.e. partial arguments) and so cooperation grants arguments to achieve knowledge completeness. The declarative semantics for Argumentation-based Negotiation is composed by argumentation and cooperation. The former imposes that every agent should argue with other agents to evaluate its knowledge. The latter allows an agent to handle its incomplete knowledge with the 'help' of other agents. Therefore, each agent α has both sets of argumentative and cooperative agents — $\mathcal{A}rgue_\alpha$ and $\mathcal{C}ooperate_\alpha$, respectively — and α must reach a consensus of its arguments with agents in $\mathcal{A}rgue_\alpha$ and

α may ask for arguments from agents in $\mathcal{C}ooperate_\alpha$ to complete its partial arguments.

We introduce cooperation by defining *available arguments*: (1) every (complete or partial) argument of α is considered an available argument of α; (2) if a partial argument for L of α may be further completed with arguments from $\mathcal{C}ooperate_\alpha$, this further completed argument is also an available argument of α. Furthermore, an agent in a multi-agent setting $\mathcal{A}$ may be able to build a (partial or complete) argument for any L in $\mathcal{H}(\mathcal{A})$ even though α has no knowledge about such an L: it will depend upon the arguments from $\mathcal{C}ooperate_\alpha$. We further propose "indirect cooperation": it may occur between argumentative agents when a proposed argument A can be used to complete a partial opposing argument B, and so the resulting argument from $A + B$ is used to attack A.

We propose the idea that every argument A of an agent α can be used in a cooperation process if and only if A is initially evaluated by $\mathcal{A}rgue_\alpha$, i.e. cooperation and argumentation are interlaced processes. As with other argumentation based frameworks the semantics is defined based on a notion of acceptable arguments, where a set of arguments is acceptable if any argument attacking it is itself attacked by the set. In this distributed setting, we had to define which arguments can be used to attack a set of arguments, and which arguments are available for being used to defend the attacks. We define that for a given agent α in a multi-agent setting $\mathcal{A}$, an agent $\beta \in \mathcal{C}ooperate_\alpha$ cooperates with an available argument A under one of the following conditions: (i) A is not attacked by any argument from $\mathcal{A}rgue_\beta$, or (ii) A is attacked, but every attacking argument B against A is attacked by some argument from $\mathcal{A}rgue_\beta$. We propose (ii) by considering that $\mathcal{A}rgue_\alpha$ can evaluate A as defensible, and such an argument might be evaluated as justified by other set of argumentative agents. Therefore, an agent cooperates with both justified and defensible arguments.

Any agent in an MAS can be queried regarding the truth value of a conclusion. Moreover, the truth value of an agent's belief depends on from which agent such a belief is inferred, and also on the specification of both sets $\mathcal{A}rgue_\alpha$ and $\mathcal{C}ooperate_\alpha$. Nevertheless, such answer is always consistent/paraconsistent with the knowledge base of the involved agents. Assuming that every agent argues and cooperates with all agents in an argumentation-based negotiation process, the results from such a process and the argumentation-based process (over the set of all agent's knowledge base coincide (cf. proof of Theorem 51). However, our proposal allows modelling a multi-agent setting with different kinds of representation, such as when a multi-agent setting stands for a kind of hierarchy of knowledge found in an organization, where

each agent has only partial knowledge of the overall process (see Examples 26 and 36); or when each agent represents "acquired knowledge" in a different period of the time and we want to know the truth value of some agent's belief in a specific period of time (see Examples 37 and 38).

We improve the idea of 'automated negotiation' by proposing a negotiation without a 'meta-agent' that controls and evaluates the overall negotiation process. Moreover, we propose an architecture to be implemented over a reliable network communication layer. The network communication layer permits the agents communication in a LAN or WAN network, the delivery of messages to every involved agents, and new agents or crashed agents are handled in the sets of argumentative and cooperative agents. It provides a fault tolerance in a negotiation process, i.e. such a process can continue if an agent crashes. In other words, if an agent Ag_1 knows that an involved agent Ag_2 crashed, Ag_1 will not wait for any answer from Ag_2 and will continue its (argumentative or cooperative) process by considering only the active agents. This is important because it releases an agent from a pending answer from a crashed agent, thus avoiding deadlocks.

6.1 Future Work

During the work performed in the preparation of this dissertation, namely in its final period, we were able to identify a set of open research aspects that we plan to tackle in the future and that will be described in this section.

Sets of Strict and Defeasible rules We may extend our proposal to express defeasible/strict rules. A DeLP-program [GS04]'s is denoted by a pair (Π, Δ) distinguishing the subset Π of facts and strict rules (that represent non-defeasible knowledge), and the subset Δ of defeasible rules. Intuitively, we could model (Π, Δ) by defining a multi-agent setting $\mathcal{A}$ with two agents, viz. a *strict agent* Ag_Π and a *defeasible agent* Ag_Δ. Ag_Π does not argue with any agent (not even itself if we assume that the set of facts and strict rules is non contradictory, as DeLP), and Ag_Π only cooperates with Ag_Δ with "facts". Intuitively, if Ag_Π cooperates with arguments $< \Pi, [L] >$ then no argument can attack it.

Updates It seems clear that the flexibility offered by the sets of cooperative and argumentative agents allows for giving priority to sets of rules over other sets of rules. This is somehow similar to logic programming updates. Example 38 suggests how our framework can be used for modelling updates. In this example, the results coincide with those of [ALP+00], but we need a study on how general this equivalence is.

Belief Revision We intend to introduce the capability of the agents to revise their knowledge as a consequence of internal or external events. In case of no agreement in a negotiation process, the agents would be able to discuss (or negotiate again) how and when they got their knowledge, and try to find a way to reach an agreement.

If an objective literal L leads to *falsity*, L is also contradictory because there is an argument A_L such that $A_\perp$ is built based on such an argument, e.g. $A_\perp : A_L + [\perp \leftarrow L, not\ L']$. However, we should point out that this solution may not be complete in the sense that we only consider the conclusions directly involved with *falsity* (cf. Def. 40). It would be better to define a new status, e.g. *generate contradiction*, for the conclusions that are involved with any case of contradiction. However, to define this new status is as complex as defining a method of "belief revision", e.g. [DP97, DPS97].

Solving conflicts among Object Constraint Language's restrictions

The Unified Modelling Language (UML) is designed specifically to represent object-oriented (OO) systems (for details see [Pen03]). Object-oriented development techniques describe software as a set of cooperating blocks of information and behavior. The standardized architecture of UML is based on the Meta Object Facility (MOF) Core [OMG06]. The MOF defines standard formats for the key elements of a model so that they can be stored in a common repository and exchanged between modelling tools, e.g. UModel2005 [Alt], Together2006 [Bor], Rational [IBM], Objecteering [SOF], and System Architect [SPA].

The Object Constraint Language (OCL) [WK] is a notational language for analysis and design of software systems, which is used in conjunction with UML to specify the semantics of the building blocks precisely. OCL has been defined to impose restrictions over UML's diagrams. OCL can be used both at the UML metamodel and model levels. At the metamodel level if is used to specify well-formedness rules, that is, rules that must be complied to by models built upon the specified metamodel. At the model level it can be used to specify domain constraints (e.g. contract rules).

The meta-model of UML has been used by the Object Management Group (OMG) as the standard for Software Engineering support tools. Support tools are aimed at making UML easier to use such as UModel2005, Rational and Objecteering. However, these support tools still accept an OCL expression as 'simple text', i.e. without validating it. Nevertheless, both support tools Together2006 and System Architect validate OCL expressions. Furthermore, there are some special tools to validate OCL expressions, for instance IBM OCL Parser v0.31 , OCLE 2 , and USE3.

The above checking tools do not verify conflicts between restrictions. By conflicting restrictions we mean that a conflict might occur because two restrictions cannot be true at the same time, or some restriction is concluded as true and so it causes that other one will be false. Furthermore, restrictions might be conflicting in a domain's modelling level, but also in both metal-level of UML and OCL. In the following we illustrate a situation of conflicting restrictions:

> "Call processing systems are large, distributed systems. Traditionally, they were either telephone exchanges or private branch exchanges under the control of a single provider or owner. Recent developments have shifted the whole area of call processing towards an open market and a more unified view of communications, considered more than Plain Old Telephone System (POTS) calls. Furthermore, the functionality has been largely enhanced by the provisioning of features. Features are extensions to the basic service, developed independently of the basic service. The last developments allow operators/providers to develop their own features. With the growing number of features that might be developed by different providers, a problem known as 'feature interaction' gains importance. Features that work independently as expected, cause some unexpected/undesired behavior when combined in a system. A typical example is a user subscribing to a Call Forwarding on Busy feature (which redirects incoming calls to a busy line to a different phone) and a Call Waiting feature (which plays a tone when a call comes into a busy line). Assume the user is busy and an incoming call arrives. This raises the question of which of the feature's behavior should be activated, as both together do not provide sensible behavior..." [RMT]

Based on such a description we are motivated to solve the following questions:

- How can conflicting restrictions be detected?
- How to solve conflicting restrictions? Can we determine which restriction should be preferred?

Therefore, the main goal would be to solve conflicts between OCL restrictions by dynamically determining preferences among such restrictions. The aim would be to develop a comprehensive approach to the engineering of software systems for service-oriented overlay computers where foundational theories, techniques and methods are fully integrated in a software engineering approach.

Bibliography

[AC02] Leila Amgoud and Claudette Cayrol. A reasoning model based on the production of acceptable arguments. *Ann. Math. Artif. Intell.*, 34(1-3):197–215, 2002.

[ADP95] José Júlio Alferes, Carlos Viegas Damásio, and Luís Moniz Pereira. A logic programming system for nonmonotonic reasoning. *J. Autom. Reasoning*, 14(1):93–147, 1995.

[ALP+00] José Júlio Alferes, João Alexandre Leite, Luís Moniz Pereira, Halina Przymusinska, and Teodor C. Przymusinski. Dynamic updates of non-monotonic knowledge bases. *J. Log. Program.*, 45(1-3):43–70, 2000.

[Alt] Altanova. *Umodel 2005*. Available at http://www.altanova.com.

[AP96] José Júlio Alferes and Luís Moniz Pereira. *Reasoning with Logic Programming*, volume 1111 of *Lecture Notes in Computer Science*. Springer, 1996.

[APS04] José Júlio Alferes, Luís Moniz Pereira, and Terrance Swift. Abduction in well-founded semantics and generalized stable models via tabled dual programs. *TPLP*, 4(4):383–428, 2004.

[arg10] Argument and computation ejournal. Taylor and Francis, 2010. Available at http://www.tandf.co.uk/journals/tarc.

[AV09] Leila Amgoud and Srdjan Vesic. Repairing preference-based argumentation frameworks. In Craig Boutilier, editor, *IJCAI*, pages 665–670, 2009.

[BC03] Trevor J. M. Bench-Capon. Persuasion in practical argument using value-based argumentation frameworks. *J. Log. Comput.*, 13(3):429–448, 2003.

[BCA09] Trevor Bench-Capon and Katie Atkinson. *Argumentation in Artificial Intelligence*, chapter Abstract Argumentation and Values. In Rahwan and Simari [RS09], 2009.

[BDKT97] Andrei Bondarenko, Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artif. Intell.*, 93:63–101, 1997.

[Bea02] B. Ben and et al. Jgroups – a toolkit for realiable multicast communication. Technical report, Jgroups project, 2002. Toolkit available at http://www.jgroups.org.

[BG09] Pietro Baroni and Massimiliano Giacomin. *Argumentation in Artificial Intelligence*, chapter Semantics of Abstract Argument Systems. In Rahwan and Simari [RS09], 2009.

[Bor] Borland. *Together 2006 for eclipse.* Available at www.borland.com/us/products/together/.

[Cal04] Miguel Calejo. Interprolog: Towards a declarative embedding of logic programming in java. In José Júlio Alferes and João Alexandre Leite, editors, *JELIA*, volume 3229 of *Lecture Notes in Computer Science*, pages 714–717. Springer, 2004. Toolkit available at http://www.declarativa.com/InterProlog/.

[CMDM05] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Prudent semantics for argumentation frameworks. In *ICTAI*, pages 568–572. IEEE Computer Society, 2005.

[CML00] Carlos Iván Chesñevar, Ana Gabriela Maguitman, and Ronald Prescott Loui. Logical models of argument. *ACM Comput. Surv.*, 32(4):337–383, 2000.

[CS05] Carlos Iván Chesñevar and Guillermo R. Simari. A tutorial on computational models for argumentation in mas, 2005. in the EASSS 2005 (Utrecht, NL). Available at http://www.cs.uns.edu.ar/g̃rs/Publications/index-publications.html.

[CW96] W. Chen and D. S. Warren. Tabled evaluation with delaying for general logic programs. *Journal of ACM*, 43(1):20–34, 1996.

[dAA06] Iara Carnevale de Almeida and José Júlio Alferes. An argumentation-based negotiation framework. In K. Inoue, K. Satoh, and F Toni, editors, *VII International Workshop on Computational Logic in Multi-*

agent Systems (CLIMA), volume 4371 of *LNAI*, pages 191–210. Springer, 2006. Revised Selected and Invited Papers.

[dAMA97] Iara de Almeida Móra and José Júlio Alferes. Credulous and sceptical argumentation for extended logic programming. Technical report, CENTRIA, Universidade Nova de Lisboa, Portugal, 1997.

[dAMA98a] Iara de Almeida Móra and José Júlio Alferes. Argumentation and cooperation for distributed extended logic programs. In Juergen Dix and Jorge Lobo, editors, *7th International Workshop on Nonmonotonic Reasoning (NMRW)*, 1998.

[dAMA98b] Iara de Almeida Móra and José Júlio Alferes. Argumentative and cooperative multi-agent system for extended logic programming. In Flávio Moreira de Oliveira, editor, *SBIA*, volume 1515 of *Lecture Notes in Computer Science*, pages 161–170. Springer, 1998.

[dAMA99] Iara de Almeida Móra and José Júlio Alferes. Conflict resolution between argumentative agents. Technical report, CENTRIA, Universidade Nova de Lisboa, Portugal, 1999.

[dAMAS97] Iara de Almeida Móra, José Júlio Alferes, and Michael Schroeder. Argumentation for distributed extended logic programs. In *Multi Agents and Logic Programming*, Leuven, Belgium, 1997.

[DMT02a] Phan Minh Dung, Paolo Mancarella, and Francesca Toni. Argumentation-based proof procedures for credulous and sceptical non-monotonic reasoning. In *Computational Logic: Logic Programming and Beyond*, pages 289–310, 2002.

[DMT02b] Phan Minh Dung, Paolo Mancarella, and Francesca Toni. Argumentation-based proof procedures for credulous and sceptical non-monotonic reasoning. In Antonis C. Kakas and Fariba Sadri, editors, *Computational Logic: Logic Programming and Beyond*, volume 2408 of *Lecture Notes in Computer Science*, pages 289–310. Springer, 2002.

[DMT06] Phan Minh Dung, Paolo Mancarella, and Francesca Toni. A dialectic procedure for sceptical, assumption-based argumentation. In Paul E. Dunne and Trevor J. M. Bench-Capon, editors, *COMMA*, volume 144 of *Frontiers in Artificial Intelligence and Applications*, pages 145–156. IOS Press, 2006.

[DP97] Carlos Viegas Damásio and Luís Moniz Pereira. A paraconsistent semantics with contradiction support detection. In Jürgen Dix, Ulrich Furbach, and Anil Nerode, editors, *LPNMR*, volume 1265 of *Lecture Notes in Computer Science*, pages 224–243. Springer, 1997.

[DPP98] Jürgen Dix, Luís Moniz Pereira, and Teodor C. Przymusinski, editors. *Logic Programming and Knowledge Representation, Third International Workshop, LPKR '97, Port Jefferson, New York, USA, October 17, 1997, Selected Papers*, volume 1471 of *Lecture Notes in Computer Science*. Springer, 1998.

[DPS97] Carlos Viegas Damásio, Luís Moniz Pereira, and Michael Schroeder. Revise: Logic programming and diagnosis. In Jürgen Dix, Ulrich Furbach, and Anil Nerode, editors, *LPNMR*, volume 1265 of *Lecture Notes in Computer Science*, pages 354–363. Springer, 1997.

[Dun93] Phan Minh Dung. An argumentation semantics for logic programming with explicit negation. In *ICLP*, pages 616–630, 1993.

[Dun95] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.

[EGH95] Morten Elvang-Gøransson and Anthony Hunter. Argumentative logics: Reasoning with classically inconsistent information. *Data Knowl. Eng.*, 16(2):125–145, 1995.

[FKIS09] Marcelo A. Falappa, Gabriele Kern-Isberner, and Guilhermo R. Simari. *Argumentation in Artificial Intelligence*, chapter Belief Revision and Argumentation Theory. In Rahwan and Simari [RS09], 2009.

[Gar00] Alejandro Javier García. *Defeasible Logic Programming: Definition, Operational Semantics and Paralelism*. PhD thesis, Computer Science and Engineering Department of Universidad Nacional del Sur, Bahía Blanca, Argentina, 2000.

[GDS09] Alejandro Javier García, Jürgen Dix, and Guillermo Ricardo Simari. *Argumentation in Artificial Intelligence*, chapter Argument-based Logic Programming. In Rahwan and Simari [RS09], 2009.

[GL90] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In Warren and Szeredi, editors, *7th International Conference on LP (ICLP)*, pages 579–597. MIT Press, 1990.

[GS04] Alejandro Javier García and Guillermo Ricardo Simari. Defeasible logic programming: An argumentative approach. *TPLP*, 4(1-2):95–138, 2004.

[IBM] IBM. *Rational software architecture an modeler.* Available at http://www-306.ibm.com/software/awdtools/developer/rose/.

[JFJ+96] Nicholas R. Jennings, Peyman Faratin, M. J. Johnson, Timothy J. Norman, P. O'Brien, and M. E. Wiegand. Agent-based business process management. *Int. J. Cooperative Inf. Syst.*, 5(2&3):105–130, 1996.

[Lif96] Vladimir Lifschitz. Foundations of logic programming. pages 69–127, 1996.

[Lou87] R. P. Loui. Defeat among arguments: a system of defeasible inference. *Journal of Computational Intelligence*, 3(2):100–106, 1987.

[Lou98] R. P. Loui. Process and policy: Resource-bounded non-demonstrative reasoning. *Journal of Computational Intelligence*, 14:1–38, May 1998.

[Moo85] R. Moore. Semantics considerations on nonmonotonic logic. *Artificial Intelligence*, (25):75–94, 1985.

[Nut94] D. N. Nute. *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3 of *Nonmonotonic Reasoning and Uncertain Reasoning*, chapter Defeasible Logic, pages 355–395. Oxford University Press, 1994.

[OMG06] OMG. *Object Management Group. Meta object facility (mof) core specification*, 2.0 edition, 2006. Document "formal/06-01-01". Available at http://www.omg.org/docs/formal/06-01-01.pdf.

[PA92] Luís Moniz Pereira and José Júlio Alferes. Well founded semantics for logic programs with explicit negation. In *ECAI*, pages 102–106, 1992.

[Pen03] Tom Pender. *UML Bible.* Wiley Publishing, Inc., 2003.

[Pol74] J. L. Pollock. Knowledge and justification. 1974. Princeton University Press.

[Pol87] John L. Pollock. Defeasible reasoning. *Cognitive Science*, 11(4):481–518, 1987.

[Pol91] J. L. Pollock. A theory of defeasible reasoning. *International Journal of Intelligent Systems*, 6:33–54, 1991.

[Pol92] J. L. Pollock. How to reason defeasibly. *Journal of Artificial Intelligence*, 57(1):1–42, September 1992.

[Pol95] J. L. Pollock. *Cognitive Carpentry: A Blueprint for How to Build a Person*. MIT Press, 1995.

[Pol96] John L. Pollock. A general-purpose defeasible reasoner. *Journal of Applied Non-Classical Logics*, 6(1), 1996.

[Pol01] John L. Pollock. Defeasible reasoning with variable degrees of justification. *Artif. Intell.*, 133(1-2):233–282, 2001.

[PP90] H. Przymusinska and T. Przymusinski. *Semantic issues in deductive databases and logic programs*, pages 321–367. 1990.

[Pra93] Henry Prakken. An argumentation framework in default logic. *Ann. Math. Artif. Intell.*, 9(1-2):93–132, 1993.

[Pra09] Henry Prakken. An abstract framework for argumentation with structured arguments. Technical report, Department of Information and Computing Sciences, Utrecht University, 2009. Available at http://www.cs.uu.nl/research/techreps/repo/CS-2009/2009-019.pdf.

[Prz90] T. Przymusinski. Extended stable semantics for normal and disjunctive programs. In Warren and Szeredi, editors, *7th International Conference on Logic Programming (ICLP)*, pages 459–477. MIT Press, 1990.

[PS97] Henry Prakken and Giovanni Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 7(1), 1997.

[PSJ98] S. Parsons, C. Sierra, and N. R. Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computational*, 8(8):261–292, 1998.

[PV02] H. Prakken and G. A. W. Vreeswijk. *Handbook of Philosophical Logic*, volume 4, chapter Logics for Defeasible Argumentation, pages 218–319. Kluwer Academic, 2 edition, 2002.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[RMT] S. Reiff-Marganiec and K. J. Turner. Use of logic to describe enhanced communications services. In *Formal Techniques for Networked and Distributed Systems (FORTE)*.

[RRJ+04] I. Rahwan, S. D. Ramchurn, N. R. Jennings, P. McBurney, S. Parsons, and L. Sonenberg. Argumentation-based negotiation. *Journal of The Knowledge Engineering Review*, 2004.

[RS09] Iyad Rahwan and Guilhermo R. Simari, editors. *Argumentation in Artificial Intelligence*. Springer, 2009.

[Sch99] Michael Schroeder. An efficient argumentation framework for negtiating autonomous agents. In Francisco J. Garijo and Magnus Boman, editors, *MAAMAW*, volume 1647 of *Lecture Notes in Computer Science*, pages 140–149. Springer, 1999.

[SdAMA97] Michael Schroeder, Iara de Almeida Móra, and José Júlio Alferes. Vivid agents arguing about distributed extended logic programs. In Ernesto Costa and Amilcar Cardoso, editors, *Progress in Artificial Intelligence, 8th Portuguese Conference on Artificial Intelligence (EPIA)*, volume 1323 of *LNAI*, pages 217–228. Springer, 1997.

[SL92a] Guillermo Ricardo Simari and Ronald Prescott Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artif. Intell.*, 53(2-3):125–157, 1992.

[SL92b] Guillermo Ricardo Simari and Ronald Prescott Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artif. Intell.*, 53(2-3):125–157, 1992.

[SOF] SOFTEAM. *Objecteering uml enterprise edition v5.3.0*. Available at http://www.objecteering.com/.

[SPA] SPARX. *Enterprise architect's uml 2.0*. Available at http://www.sparxsystems.com/products/ea.html.

[SPR98] Michael Schroeder, Daniela Alina Plewe, and Andreas Raab. Ultima ratio: Should hamlet kill claudius? In *Agents*, pages 467–468, 1998.

[SS02a] Michael Schroeder and Ralf Schweimeier. Arguments and misunderstandings: Fuzzy unification for negotiating agents. *Electr. Notes Theor. Comput. Sci.*, 70(5), 2002.

[SS02b] Ralf Schweimeier and Michael Schroeder. Notions of attack and justified arguments for extended logic programs. In Frank van Harmelen, editor, *ECAI*, pages 536–540. IOS Press, 2002.

[SW07] Terrance Swift and David S. Warren. The xsb logic programming system. *Association for Logic Programming Newsletter*, 2007. Summary of recent develpments in XSB.

[Swi99] Terrance Swift. A new formulation of tabled resolution with delay. In Pedro Barahona and José Júlio Alferes, editors, *EPIA*, volume 1695 of *Lecture Notes in Computer Science*, pages 163–177. Springer, 1999.

[Tar55] A. Tarski. A lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[Vre93] G. A. W. Vreeswijk. *Studies in Defeasible Argumentation*. PhD thesis, Free University Amsterdam, Departament of Computer Science, 1993.

[Vre97] G. A. W. Vreeswijk. Abstract argumentation systems. *Journal of Artificial Intelligence*, 90(1–2):225–279, 1997.

[vRS91] Allen van Gelder, Kenneth Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *ACM*, 38(3):620–650, 1991.

[WK] J. B. Warmer and A. G. Klepe. *The Object Constraint Language: precise modelling with UML*. Addison-Wesley, 2 edition.