A TEMPORAL ARGUMENTATION THEORY FOR MEDICAL DIAGNOSIS

SÉLINDE VAN ENGELENBURG (STUD. NR.: 0309389)
SUPERVISORS: JOHN-JULES MEYER, GERARD VREESWIJK
ECTS: 30
DATE: 24 FEBRUARY 2012

CONTENT

1 INTRODUCTION

Medical artificial Intelligence has its origins in the early 1970's. Research on the application of artificial intelligence in the medical domain resulted in the first clinical decision support systems such as Internist-1, CASNET and MYCIN in the mid 1970's and early 1980's (Patel, et al., 2009). Nowadays, interest in clinical decision support systems is still growing and they become widespread. (Berner & La Lande, 2007). Clinical support systems are systems purposed to improve clinical decision making by matching characteristics of patients to a knowledgebase and generating patient-specific recommendations.(Garg, et al., 2005). Clinical decision support systems can aid clinicians in several different ways. In their extensive review of clinical decision support systems, (Garg, et al., 2005) distinguish four different types, systems for diagnosis, reminder systems for prevention, systems for disease management and systems for drug dosing and drug prescribing. According to (Garg, et al., 2005), 64% of these systems improve the performance of clinicians significantly; suggesting that the usage of these systems is helpful in the majority of cases.

In this thesis, the focus is on clinical decision support systems that aid clinicians in making diagnoses. It is investigated which things are important for diagnosing patients correctly and a logic is proposed on which a medical diagnostic system can be based.

Research for this thesis started out by an attempt to answer the following research questions:

- What are the requirements for a medical diagnostic system?
- On what logic should a medical diagnostic system be based to meet these requirements?

First, requirements for a medical diagnostic system were established as an answer to the first research question. During the efforts made to answer the second research question, the question arose whether there is in fact an existing logic that can be incorporated by a medical diagnostic system such that all requirements are met. Since no such logic could be found, the focus of research shifted to designing such a logic, resulting in the temporal argumentation logic proposed in this thesis. It was also investigated whether this proposed logic does meet the requirements and to what extent. The consequence of this shift of focus is that the emphasis of this thesis is more on the defining of the proposed logic and its advantages and disadvantages, than on answering the originally established research questions. It is in my opinion that the proposed argumentation logic and its evaluation are more important results, than the answer to the original research questions. The research questions above have been investigated but are not discussed elaborately; this would lead to a thesis incorporating too many subjects and thus not being directed enough. A short summary of the output of the investigations conducted in order to answer the second research question above is however supplemented in an appendix. The requirements established to answer the first research question are used to evaluate the proposed logic.

The investigation of the first research question was carried out by studying literature on the diagnostic process as performed by human diagnosticians and by consulting domain experts. It was determined that amongst others the ability to handle missing, contradictory and temporal information and being able to give a clear account of the reasons for making certain diagnoses, are essential for a medical diagnostic system. To answer the second research question, several kinds of logics were studied and for each it was established whether a system incorporating them would meet all requirements. Argumentation logic emerged as the logic that would satisfy most of them. Systems based on argumentation logic would be able to handle missing information, such as missing facts about a patient. In addition, they would be very well equipped to handle contradictions. It is also possible to incorporate temporal arguments in argumentation logic and they thus are equipped to handle temporal information. Furthermore, argumentation logics are recognized for being intuitive and being able to give a clear account of their reasoning process.

During consultations with experts in the medical domain, it became clear that in practice clinicians often have nothing more than partially missing or imprecise information about the temporal aspects of symptoms and other attributes of their patients. No logic could be found which can meet the requirements stated above and would be able to handle partially missing or imprecise temporal information. This led to the development of a temporal argumentation logic in which partially missing temporal information can be expressed and reasoned with. It was chosen to base this proposed logic on the argumentation logic DeLP (García & Simari, 2004). In DeLP, argumentation logic is combined with Logic Programming and thus it is very suitable to be implemented and to be used as a basis for a reasoning system. The temporal argumentation logic proposed in this thesis is defined formally and is discussed informally as well. For each of the requirements established, it is determined whether they are met by a system incorporating the proposed logic and to which extent. To determine its feasibility a proof of concept implementation of the proposed logic was made.

It is important to note that though the medical information about diseases in this thesis came from reliable sources, the standards of a medical paper are not met. This thesis is primary an artificial intelligence thesis and not a medical one. Medical information on diseases has the sole purpose of clarifying certain ideas and concepts. I am not a medical expert and this thesis should therefore not be used as a resource of information on specific diseases and medical conditions.

In section 2, the requirements of a medical diagnostic system are discussed and determined. In the subsequent section, argumentation logic in general and DeLP in particular are discussed. In section 4, the proposed temporal argumentation logic is defined formally and discussed. The proof of concept implementation of the proposed logic is discussed in section 5. In the next section, it is evaluated to which extent the requirements of section 2 are met by a system implementing the proposed logic. In section 7, additional advantages and disadvantages of the proposed logic are discussed and suggestions for further research are made. A short summary of the evaluations of the logics of which the suitability has been investigated is included in the appendix.

2 REQUIREMENTS FOR A MEDICAL DIAGNOSIS PROGRAM

Diagnosing patients is in essence inferring which disease they have on basis of information about them and their symptoms. The most straightforward way to do this in an automated reasoning program is by using the language of propositional logic to make rules of the form $fact_1 \wedge fact_2 \wedge \ldots \wedge fact_n \rightarrow disease$, where $fact_1, fact_2, \ldots, fact_n$ denote facts about patients such as their symptoms and test results and $disease$ denotes a disease. The program could search for a rule for which the antecedent matches the facts that are known about a patient. If such a rule is found, the program can derive the consequent and establish that the patient has the disease denoted in it.

The medical diagnosis program described above does have several advantages. Diagnosing the patient would be quick, since the only thing that needs to be done is to find the appropriate rule based on the facts about the patient and deriving its consequent. Furthermore, it would be fairly easy to explain to a user how and why the program made a certain diagnosis. Only the rule and the facts on basis of which the rule is chosen have to be displayed to do this.

Although there are some benefits to such a simple program, there are several severe shortcomings as well. When more closely examined, diagnosis turns out to be a process that is much more complex, refined and flexible. An automated reasoning program capable of making precise diagnoses should be able to capture this and the simple program described here is clearly not. In this section, I will describe some of the necessities for accurate diagnosis. These requirements for diagnosis are mainly based on freely accessible information about the diagnostic process as performed by human diagnosticians, the consultation with domain experts and common sense. In the subsequent sections, I will provide a description of the logic I propose to use in a medical diagnostic program and I will evaluate whether this logic meets the mentioned requirements for accurate diagnosis.

2.1 INCOMPLETE INFORMATION

One of the shortcomings of the simple program is that it is assumed implicitly that all relevant facts about the patient are known. In practice, this is hardly ever the case. Patients may for example neglect to report some of the symptoms they experience and some non-directly observable properties of the patient, such as blood levels may not be known. Patients could of course be asked whether they experience certain additional symptoms and supplementary tests can be done to measure non-directly observable properties. However, it is not feasible to ask the patient for every possible symptom and to do every possible test. Consequently, we still cannot assume we know every relevant fact about the patient. There are several ways to deal with this, whether diagnoses are made by a human physician or a computer.

The usual strategy doctors use to make a diagnosis depends on hypothesis generation and hypothesis testing. To generate hypotheses, the main diagnostic possibilities based on the known facts about the patient are identified. This identification of the diagnostic possibilities, or differential diagnoses, is usually done by pattern recognition. Following, each differential diagnosis gets an estimated likelihood assigned. Based upon, amongst others, these estimations of likelihood, doctors try to obtain more information to refute or support certain possibilities. This information is acquired by examining patients and asking them specific questions. Then a presumptive diagnosis is made and diagnostic tests are done to reduce remaining uncertainty. Subsequently, the decision for treatment is made depending on the benefit of treating a sick person and the risk of mistakenly treating a person that does not have the disorder. (The Merck Manual, 2010).

The strategy used by physicians could be implemented in an automated reasoning system. In that case, missing data is dealt with by computing their informative value. The data which is most informative, i.e. which can be used to refute or support the most possibilities, will be requested from the user. The user can then perform the necessary tests, examinations and ask the patient questions to obtain the required information. The new data can be added to the database and the inference process can continue. The system described copies part of the strategy of physicians and will probably deal with missing information effectively. There are however cases in which this kind of system is used and when it is impossible to obtain new data. This may be because the user was not able to obtain it, or because it is in general not possible to obtain new data about the patient. An example of this is a program that is used for scientific research and has to diagnose an entire database of people. In those cases, each unknown fact about a patient will be assigned a default value, the value it is most likely to have when no information about it is available. These default values differ from normal values in that they may be changed when they become known.

As a descriptive example of the above, assume that a program contains a rule with 'high blood pressure' in its antecedent. Assume also that it is unknown whether a patient has a high blood pressure and this fact cannot be derived from any other facts known about the patient. Assume in addition that it is not possible to obtain more information about the patient's blood pressure at this time. In this case, 'high blood pressure' will get the default value 'false' assigned because people usually do not have a high blood pressure. Assigning the value 'false' to a variable when the program is not able to prove that it is true, is also called negation as failure. If later on in the diagnostic process it becomes known that the patient has a high blood pressure, the value of the variable will be changed to 'true' and new inferences can be made, while others get refuted.

2.2 CONTRADICTIONS

Contradiction arises when two facts about a patient (i.e. diseases, symptoms and such) are derived that cannot both be true. This problem may occur more frequently when the set of diagnostic rules is large and much data about patients is available. Contradictions may be due to mistakes such as false-negative or false-positive diagnostic tests and mistakes in entering patients' data in the database. Another cause of problems with the data about patients is lack of information about certain aspects of the data. When for instance temporal information about symptoms is not taken into account, it may happen that a symptom and its negation are both entered in a database. Both could have been true, but at different times. When temporal information is not reckoned with, this may lead to a contradiction. This is merely one of the reasons why temporal information is important. In one of the subsections below, I will discuss further reasons for taking temporal information into account.

Another cause of contradictions may be that the rules for making certain diagnoses are conflicted. This in turn may be due to mistakes in determining what the rules should be. In some cases, contradictions may be a result of disagreement between medical experts about the nature of diseases. It is for instance known that because of this, contradictions may be derived when using rules from the DSM-IV-TR, the handbook for psychiatrists and psychologists with the standard criteria for classifying mental disorders.(Gartner, Swift, Tien, Damásio, & Pereira, 2000)

It is vital that contradictions are noticed and dealt with since when a contradiction is derived, there is uncertainty about the diagnosis a patient should have or error could have been made. This means that a wrong diagnosis could have been made and in the medical domain this may have catastrophic consequences.

In the case of errors in the patient data, the data should be corrected. This can for example be done by correcting mistakes made when entering the data in the program or by redoing tests. The program may also be modified to take additional aspects of the data into account. When two rules are conflicting, one of them has to be rejected or adapted. In the case that the contradiction arises from disagreement in the medical domain, it may be best to let the choice of rule depend upon the preference of the attending physician. An additional solution is to specify under which conditions certain rules or derived data may be rejected or an order of precedence may be specified on rules, this may make it possible to choose automatically the more probable of two contradictory alternatives.

2.3 SINGLE OR MULTIPLE DISEASES

In some cases, it may be possible to diagnose a patient with two diseases. Often this will be no problem because the patient has in fact both diseases. However, if the two diseases diagnosed are very similar, in other words, if they share many symptoms, this may indicate otherwise. Clearly, it is possible that the patient still actually has both diseases, but it may also be the case that there is a problem distinguishing the diseases from each other.

Assume that a computer program (or a physician) uses the following rules for diagnosing $disease_1$ and $disease_2$:

Rule 1. $fact_1 \wedge \ldots \wedge fact_n \rightarrow disease_1$
Rule 2. $fact_1 \wedge \ldots \wedge fact_n \wedge fact_{n+1} \wedge \ldots \wedge fact_m \rightarrow disease_2$

In addition, assume that $fact_1 \wedge \ldots \wedge fact_n$ and $fact_{n+1} \wedge \ldots \wedge fact_m$ are known to be true for a patient. Assume also that $disease_1$ and $disease_2$ are two different diseases and that Rule 1 and Rule 2 are not two alternative ways of diagnosing it. Now both $disease_1$ and $disease_2$ could be diagnosed. This allows for three relevant possibilities, the patient has in fact both $disease_1$ and $disease_2$ or the patient only has $disease_2$ or the patient has $disease_1$ and $fact_{n+1} \wedge \ldots \wedge fact_m$ are due to other causes. Another possibility is of course that the patient has neither disease. In that case, the facts are accounted for by one or more other diseases for which there are other rules or the disease the patient has is not known to the program or to the physician. In the first case, we have the same problem as discussed in this section, only the problem ranges over more rules. In the second case the problem is due to an incomplete knowledge about diseases, which is not of concern here. Therefore, and for the sake of clarity, this possibility is ignored here.

To make an accurate diagnosis, a choice has to be made between the three remaining possibilities. The probabilities of the possibilities are likely to be different and may or may not be known. If they are not exactly known, one could for example argue that it is most probable that the patient has $disease_2$ because Rule 2 is more specific and accounts for more of the facts. On the other hand, if $disease_1$ is far more common than $disease_2$, then it may be preferable to diagnose $disease_1$. When $disease_1$ and $disease_2$ often occur together, the choice may be on diagnosing them both. Rules for which rule is preferred in certain cases can be implemented in the program or used by the physician.

In some cases, it may not be possible or desirable to make a choice. Absolute certainty may be required when for instance, one of the diseases is very serious or treatment for one of the diseases involves high risks. In this situation, a way needs to be found to derive more facts about the patient and the possible diseases. If this is absolutely impossible, no diagnosis can be made. Treatment for one or both diseases should then be chosen such that the risks for the patient are minimal.

2.4 EXCLUSIONARY CRITERIA

Rules used by an automated reasoning program or a physician may not be applicable under certain circumstances. If such a circumstance occurs frequently and excludes only a small number of diagnoses, then this could be included in the antecedent of rules, making it impossible to derive their consequents when the circumstance occurs. There are situations in which rules are excluded because of reasons less common. An example is a patient who had a cholecystectomy (removal of the gall bladder), who experiences a combination of symptoms that could indicate several diseases, including cholecystitis (inflammation of the gall bladder). However, the patient cannot possibly have cholecystitis, since the gall bladder has been removed. (The Merck Manual, 2007 (1))

In cases when the reason for the inapplicability of a rule is uncommon, it is not feasible to include it in the antecedent of the rule. There may be a great number of such circumstances and checking whether every single one is true for each disease considered is an unreasonable amount of work for a physician or program. In the case of the example above, it may thus not be feasible to include in the antecedent of the rule for diagnosing cholecystitis that no cholecystectomy has been performed, even though cholecystitis should not be diagnosed when the patient has undergone this procedure.

One method to take into account exclusionary criteria is by defining rules that specify under which circumstances other rules are inapplicable. They can be used in the case it is known that an exclusionary criterion is true, to exclude the appropriate rules. If no information about an exclusionary criterion is available, it is by default considered false (see negation as failure and default values in sec. 2.1).

2.5 TEMPORAL INFORMATION

As an illustration of the importance of temporal information for diagnosis in de medical domain, I will start by describing the example of Reye's syndrome.

Reye's syndrome is a fairly rare disease, but it is in many cases very serious. Fatality rates are between <2% and 80%, depending on the severity of the occurrence of the disease. The mean fatality rate is 21% and 30% of patients who do survive the disease will suffer from neurological sequelae such as intellectual disability and seizure disorders. This means that it is important to diagnose this disease quickly, yet Reye's syndrome is hard to diagnose because its directly observable symptoms are very common in other diseases. (The Merck Manual, 2009).

Reye's syndrome is a biphasic disease. Initial symptoms of a viral infection are followed in 5 to 7 days by nausea, vomiting and a sudden change in mental status. The change in mental status differs in severity from patient to patient, but may progress rapidly into a deep coma. The chance of developing this disease following a viral infection increases a 35-fold when salicylates, such as aspirin (The Merck Manual, 2007 (2)) are taken during the viral infection. The syndrome almost exclusively occurs in children younger than 18 years. (The Merck Manual, 2009)

From the description of Reye's syndrome, it is evident that temporal information is important. Because the symptoms are common, being able to take temporal information into account would make it easier to differentiate the diagnosis of Reye's syndrome from other diseases. The fact that patients have a viral infection 5 to 7 days prior to the other symptoms and that they took salicylates during the period they suffered from the viral infection are characteristic of Reye's syndrome. The temporal information that should be taken into account in this case would thus be the order of symptoms and events and the length of the period between symptoms.

There are more types of temporal information that may be important for making an accurate diagnosis and to differentiate between diseases. One of them is the length of time a symptom or disease has been present. To show an example of this, I will describe idiopathic interstitial pneumonia (IIP). IIPs are interstitial lung diseases of unknown cause, which share similar features such as dyspnea (shortness of breath). The diseases can be classified in six different subtypes. Each subtype is characterized by varying degrees of inflammation and fibrosis (scarring of the lungs). Identifying which subtype patients have is of importance since the choice of treatment and prognosis depend on it. (The Merck Manual, 2008 (2))

To determine the subtype of the occurrence of IIP in patients, information about their histories is important. Especially significant is information about the duration of symptoms, family history, history of tobacco use, current and prior drug use and possible exposure to pollutants in the home and work environment. For the subtype acute interstitial pneumonia it is typical that symptoms have a sudden onset and increase in severity over 7 to 14 days. On the other hand, patients suffering from the subtype idiopathic pulmonary fibrosis usually develop symptoms over more than 6 months. (The Merck Manual, 2008 (2)) To differentiate between the two, reasoning with information about the duration of symptoms may be important.

It also is possible that patients suffered from different diseases at different times. If no temporal information is taken into account, it is harder to determine whether symptoms are due to one disease or multiple ones. If symptoms are clustered in one or more groups in different periods, this may be an indication that they originate from different diseases at different times (see section 2.3). Temporal information may also be useful for determining in which stage or phase a disease is and for determining whether a disease is chronic or recurrent.

While temporal information may be vital for diagnosing a patient, in practice often there is not any, incomplete or imprecise temporal information available according to domain experts. In some cases it may for instance happen that it was not tracked when certain measurements (e.g. blood pressure, heart rate) where taken. Since such measurements generally provide only information about the instance the measurement was taken, information about the interval in which the measured symptom was present may also be missing. This may in addition occur in cases where patients do not remember exactly when and how long they experienced certain symptoms or in cases they are unable to report them. Temporal information may also be incomplete when for a diagnosis a symptom has to be present longer than the time that has elapsed since the symptom begun until “now”. A closely related cause of incomplete temporal information is that according to domain experts patients are very imprecise in reporting temporal aspects about their symptoms.

2.6 CHAINING RULES

One of the symptoms of Reye’s syndrome described in the section above is that the patient has had a viral infection, which is a disease itself. This implies that a diagnostic program being able to diagnose illnesses such as Reye’s syndrome should be able to use other diseases as a symptom of a disease. Of course, it is possible to add the symptoms of some disease to the symptoms of another disease. However, this would make a program less efficient and more redundant. This has in addition the disadvantage of making it harder to explain to the user why a certain disease was diagnosed by the program. Likewise, in some cases a combination of symptoms may be captured by a single symptom. An example of this is recurrent abdominal pain. When patients suffer for at least 3 months of intermittent abdominal pain, it is said that they have recurrent abdominal pain (The Merck Manual, 2008 (1)). Reasoning with the symptom ‘recurrent abdominal pain’ instead of all the different episodes of abdominal pain, avoids the same disadvantages as the ones described above.

2.7 CLARIFYING THE REASONING PROCESS TO USERS

It is important for the user of a medical diagnostic program to know on basis of which information a certain diagnosis was made. Mistakes can be made at several points in the diagnostic process, due to human errors or errors in the program. This means that when using a medical diagnostic program, physicians should check whether a certain diagnosis is plausible and they should be able to check whether a diagnosis was made on the right grounds. This can only be done if they have a clear view of the reasoning process of the program and the information on which a diagnosis is based.

A related motivation for making the reasoning process clear is of a more ethical nature. Even though a computer program can make a diagnosis, the responsibility for making the correct diagnosis and choosing the right treatment lies with the physician. Physicians can only take this responsibility if they understand why the program made a certain diagnosis. In addition, a physician may need to explain the motives for making a certain diagnosis to a patient, for which insight into the reasoning process is also required.

In the next section, the argumentation logic developed by Alejandro J. García and Guillermo R. Simari is described. This argumentation logic is the foundation for the temporal argumentation logic proposed in this paper. The proposed logic is formally defined and discussed in section 4. In section 6, we will return to the requirements described in this section and discuss the extent to which they are met by a program using the proposed logic.

3 ARGUMENTATION LOGIC

One of the characteristics of classical logic is monotony. Monotony arises from the idea that a proof is a sequence of steps, starting with axioms on which inference rules may be applied to derive new sentences. These inference rules stay valid in any context. (Bochman, 2007) This means that there are no conditions under which the premises of a rule are true, but the conclusion may not be derived. To make the notion of monotony more formal, assume there are sets of premises Γ and Δ such that $\Gamma \subseteq \Delta$. Assume in addition that there is a sentence ϕ such that $\Gamma \vDash \phi$. If the logic has the property of monotonicity, then $\Delta \vDash \phi$. (Aldo, 2010)

Monotonic logics have several useful applications; however, they turn out not to be suitable for modeling and reasoning about real-world situations, as is done in medical diagnosis. When reasoning about real-world situations, it will be hardly ever the case that all information about the world is known. In addition, it is possible that certain properties of the information are unknown (such as temporal properties) or that errors are made, which can bring about contradictions. Monotonic logics are not able to deal with this in a sensible and informative way. Nonmonotonic logic however has certain strengths monotonic logic does not have which make it capable of dealing with the problems mentioned above. Contrary to monotonic logic, in nonmonotonic logic conclusions are derived tentatively and can be retracted later when for instance more information is available. It is possible in nonmonotonic systems to use negation as failure, i.e. to assume that something is not true in case the contrary cannot be proven.

In the late 1970's, nonmonotonic logic was developed by amongst others J. McCarthy, D. McDermott, J. Doyle and R. Reiter. In 1980, an issue of the *Artificial Intelligence Journal* was completely dedicated to the new field of nonmonotonic logic. (Aldo, 2010) During the mid 1980's, nonmonotonic logic got a lot of attention from the AI community. Useful applications in amongst others the philosophical and legal domain were discovered. (Chesñevar, Maguitman, & Loui, 2000) Several kinds of nonmonotonic logics were developed such as Circumscription Logic by J. McCarthy and Default Logic by R. Reiter.(Aldo, 2010)

As mentioned above, when reasoning about the real world, we usually have only partial information about it. This means assumptions about the way things are by default need to be made. Without such assumptions, it would practically be impossible to reason about the real world. Reasoning using these kinds of assumptions is part of nonmonotonic reasoning. (Bochman, 2007) To illustrate, I will describe the most well known example. In nonmonotonic logic, one could formulate the sentence "Birds fly". Contrary to classical logic, this sentence does not mean that all birds fly, but that birds are entities that fly in general, i.e. if "Tweety is a bird", then in general it can be assumed that "Tweety flies". The general sentence "Birds fly" can be given up in the light of information with a better quality, such as information that is more specific or more reliable. Thus, if there is a sentence "Tweety is a penguin", which is more specific than "Tweety is a bird", then it can be assumed that "Tweety does not fly". The rules on which the commonsense reasoning in this example is built are the following: $bird(x) \mid\!\sim fly(x)$, $bird(x) \wedge penguin(x) \not\mid\!\sim fly(x)$ and $bird(x) \wedge penguin(x) \mid\!\sim \neg fly(x)$[1]. (Schlechta, 2007)

If we look at the property of monotony, it is clear that the kind of nonmonotonic reasoning used in the example cannot possibly be done using classical logic. Consider the definition in the beginning of this chapter again, together with the rules of the example. Assume that $\Gamma = \{bird(tweety)\}$ and that $\Delta = \{bird(tweety), penguin(tweety)\}$ and that $fly(tweety)$ is denoted by ϕ. Now from premises Γ it can be derived that $fly(tweety)$. If the logic is monotonic and $\Gamma \subseteq \Delta$, it should also be derived from premises Δ, that $fly(tweety)$. This means that a consistent monotonic logic containing the first rule from the example cannot contain the second and third rule. As a result, the commonsense reasoning conducted in the example cannot be done in a monotonic logic. To be able to deal with incomplete information all defeasible rules, like the ones from the example, are needed and this is not possible while using a monotonic logic.

One type of nonmonotonic logic, developed mainly in AI research is argumentation logic (Prakken & Vreeswijk, 1998). In most nonmonotonic logics the logical consequence relation is defeasible, this is however not the case with argumentation logic. In argumentation logic a proof of a sentence, called an argument is built monotonically. Defeasibility results from the interaction between conflicting arguments. An argument can be defeated if there is another argument that conflicts in some way with it (i.e. it is a counterargument) and this argument is preferred over the first argument. In the case of the example used above, an argument can be made that "Tweety flies", because "Tweety is a bird". A counterargument can be made that "Tweety does not fly" because "Tweety is a penguin". The counterargument may be preferred because "Tweety is a penguin" is more specific than "Tweety is a bird". In that case, the first argument is defeated by the second one. (Prakken & Vreeswijk, 1998) There are several other forms of defeat possible, depending of the kind of argumentation logic used.

In their paper *Logics for Defeasible Argumentation*, Henry Prakken and Gerard Vreeswijk describe five elements of every argumentation logic. The first element they describe is an underlying logical language. Sentences in argumentation logic are expressed in this logical language. The logical consequence relation is as well part of the logical language. Like mentioned above, this logical consequence relation is monotonic and does by itself not give rise to defeasibility.

Defeasibility arises from the interaction between contradicting arguments. Arguments are described by Prakken and Vreeswijk as corresponding to monotonic proofs in the underlying logic. The sentence that is 'proven' is the conclusion of the argument. Arguments can have different forms depending on the argumentation logic in which they are defined.

Though arguments are built monotonically themselves, different arguments in the logical system may be conflicted, it is also said that arguments are attacked by other arguments or

[1] $\mid\!\sim$ is the defeasible consequence relation, x is a variable

that arguments attack each other. Prakken and Vreeswijk describe two types of attack which are usually distinguished. The first type is symmetrical and is called rebutting attack. Two arguments are said to rebut each other when they have contradictory conclusions. The second type of attack, undercutting attack is asymmetric. There are in turn two types of undercutting attack. For the first type, it is assumed in an argument that a sentence is not provable and the conclusion of this argument is derived (partly) based on this assumption. This kind of argument can be undercut by an argument with a proof of this sentence. The second type is when an argument contradicts the link between premises and a conclusion of another argument. Besides the types of conflicts described here, several other types are possible[2] as will be shown when the argumentation logic used in the medical diagnostic program is discussed.

When an argument attacks another one, it has to be determined which argument is ‘stronger’ or is preferred. To be able to evaluate a pair of conflicting arguments, certain criteria have to be set. Such a criterion could be that more specific arguments are preferred or that arguments based on fewer assumptions are preferred. Often criteria are provided by users, since the optimal criteria may be domain-specific. (Prakken & Vreeswijk, 1998) The fourth element of an argumentation logic described by Prakken and Vreeswijk is that of defeat among arguments. Defeat is essentially a binary relation on a pair of arguments which has a weak form (defeat) and a strong form (strict defeat). An argument is defeated by another argument if it is attacked by this argument and the attacking argument is not weaker as determined by the evaluation criteria. An argument is strict defeated if its attacker is stronger. (Prakken & Vreeswijk, 1998)

Since the defeat relation only specifies the relation between pairs of arguments, in addition a status assignment for arguments is needed to determine the ultimate status of arguments based on the interaction between all arguments. This definition of the status of arguments is the fifth element that Prakken and Vreeswijk discuss. Arguments may be assigned one of several possible statuses on basis of the interaction between all arguments, depending on the type of the argumentation logic. An example of an interaction between more than a pair of arguments is when the defeater B of an argument A is itself defeated by an argument C. In that case, in the status assignment it is often defined that the status of argument A is changed back from ‘defeated’ to ‘undefeated’. Changing the status of argument A is also called a reinstatement of A. It is also said that C reinstates A in this case. Another principle that is often defined in the status assignment is that if a subargument of an argument is defeated, the argument cannot be considered undefeated anymore. (Prakken & Vreeswijk, 1998)

In the medical diagnostic program, the argumentation logic DeLP in combination with temporal arguments is used. DeLP is an argumentation logic developed by Alejandro J. García and Guillermo R. Simari and described in their paper *Defeasible Logic Programming An Argumentative Approach (2004)*. DeLP stands for ‘defeasible logic programming’ and like

[2] Even types bearing the same names as the ones described here, but with different definitions.

this name suggests, it is a combination of logical programming and argumentation logic. The combination of argumentation logic makes DeLP very suitable to use as basis of a computer program. In the subsections below, the way in which each of the five elements of argumentation logic is defined in DeLP is discussed using the definitions from (García & Simari, 2004).

3.1 THE UNDERLYING LOGICAL LANGUAGE AND LOGIC PROGRAMS

García and Simari define literals in the underlying logical language to be ground atoms or their negations. The version of negation used in DeLP is strong negation and denoted by $\sim$. The language of DeLP consists of three sets. The first set is a set of facts, this set of facts consists of literals. Facts are used to represent knowledge that is immediately available, such as data from a database about the symptoms of a patient. The second set is the set of strict rules. Strict rules are used to represent knowledge that is sound and indefeasible. An example of a strict rule from the medical domain is $\sim uterus \leftarrow man$ denoting that men do not have uteruses. The third set is the set of defeasible rules. These rules are used to represent weak or tentative knowledge; this knowledge may only be used if there is no 'stronger' knowledge opposing it. An example of a defeasible rule from the medical domain could be $common\_cold \prec coughing, sore\_throat, runny\_nose$ denoting that "reasons to believe that people are coughing, have a sore throat and have a runny nose, provide reasons to believe that they have the common cold". In the definitions below, each of the sets is defined formally.

DEFINITION 3.1 (FACTS)(GARCÍA & SIMARI, 2004)

A fact is a literal, i.e. a ground atom or a negated ground atom.

DEFINITION 3.2 (STRICT RULE) (GARCÍA & SIMARI, 2004)

A strict rule is an ordered pair, denoted "$Head \leftarrow Body$" whose first member $Head$, is a literal, and whose second member, $Body$ is a finite non-empty set of literals. A strict rule with the head L_0 and body $\{L_1, \ldots, L_n\}$ can also be written as $L_0 \leftarrow L_1, \ldots, L_n$ $(n > 0)$.

DEFINITION 3.3 (DEFEASIBLE RULE) (GARCÍA & SIMARI, 2004)

A defeasible rule is an ordered pair, denoted "$Head \prec Body$" whose first member $Head$, is a literal, and whose second member, $Body$ is a finite non-empty set of literals. A defeasible rule with the head L_0 and body $\{L_1, \ldots, L_n\}$ can also be written as $L_0 \prec L_1, \ldots, L_n$ $(n > 0)$.

In logic programming, the rules defined above are called program rules. In DeLP, program rules are not allowed have an empty body. This does not mean that program rules with an empty body cannot be represented in DeLP. Strict rules with an empty body can be represented by facts. Representing defeasible rules with an empty body is somewhat a more complicated matter. Defeasible rules with an empty body are called presumptions and are different from facts. García and Simari add them as an extension to DeLP.

A defeasible logic program is a set containing facts, strict rules and defeasible rules. This set can be infinite, which deviates from the usual notion of a logic program. A distinction is made between indefeasible knowledge, i.e. facts and strict rules and defeasible knowledge, i.e. defeasible rules. In principle, strict and defeasible rules are ground. However, in the examples García and Simari provide, rules with variables are used and thus the grounded versions of defeasible logic programs are defined as the set of facts and all grounded instances of rules.

The output of a defeasible logic program is one of the following answers to a query: YES, NO, $UNDECIDED$ and $UNKNOWN$. The procedure for obtaining these answers will be discussed in subsequent sections. Below is the definition from (García & Simari, 2004) of a defeasible logic program. Successively, an example is given of a defeasible logical program and its answers to some possible queries.

DEFINITION 3.4 (DEFEASIBLE LOGIC PROGRAM) (GARCÍA & SIMARI, 2004)

A defeasible logic program $\mathcal{P}$, abbreviated de.l.p., is a possibly infinite set of facts, strict rules and defeasible rules. In a program $\mathcal{P}$, we will distinguish the subset Π of facts and strict rules and the subset Δ of defeasible rules. When required we will denote $\mathcal{P}$ as (Π, Δ).

Strict and defeasible rules are ground. However following the usual convention, some examples will use "schematic rules" with variables. Given a "schematic rule" R, $Ground(R)$ stands for the set of all ground instances of R. Given a de.l.p $\mathcal{P}$ with schematic rules, we define:

$$Ground(\mathcal{P}) = \bigcup_{R \in \mathcal{P}} Ground(R)$$

In order to distinguish variables from other elements of a schematic rule, we will denote variables with an initial uppercase letter.

In real-world situations, there may be things that are true in most cases, which can be represented by defeasible rules and there may be things that are always true, which can be represented by strict rules and facts. To give an example from the medical domain, the knowledge that most people with symptoms of sneezing and itching eyes after coming into contact with a cat are allergic to cats, can be represented by a defeasible rule. Another example from the medical domain is that if someone is dead, he or she is not alive. The knowledge from the latter example can be represented by a strict rule since it is always true.

Because such a distinction between types of knowledge about real-world situations is intuitively present, it seems natural to also make a distinction between types of rules and facts representing this knowledge. This is exactly what is done in DeLP. Some other reasons derived from a more logical and technical viewpoint for making a distinction between facts and strict rules and defeasible rules can be found below.

Below is an example of a defeasible logic program incorporating amongst others the rules from the 'Tweety'-example in the introduction of this section.

EXAMPLE 3.1 (ADAPTED FROM (GARCÍA & SIMARI, 2004))

Here follows the de.l.p. $\mathcal{P}_{3.1} = (\Pi, \Delta)$, where sets Π (strict rules and facts) and Δ (defeasible rules) have been separated for convenience of the presentation:

$$\Pi = \left\{ \begin{array}{c} bird(X) \leftarrow chicken(X), \\ bird(X) \leftarrow penguin(X), \\ {\sim}flies(X) \leftarrow penguin(X), \\ chicken(tina), \\ penguin(tweety), \\ scared(tina) \end{array} \right\}$$

$$\Delta = \left\{ \begin{array}{c} flies(X) \prec bird(X), \\ {\sim}flies(X) \prec chicken(X), \\ flies(X) \prec chicken(X), scared(X), \\ nests\_in\_trees(X) \prec flies(X) \end{array} \right\}$$

As will be shown in the following sections, in DeLP the answer for $flies(tina)$ will be YES, whereas the answer for ${\sim}flies(tina)$ will be NO. The answer for $flies(tweety)$ will be NO and the answer for ${\sim}flies(tweety)$ will be YES.

3.2 ARGUMENTS

To be able to explain the notion of an argument in DeLP, first the derivation of new literals has to be discussed. A derivation in DeLP is a sequence of literals which are facts or which are obtained by using defeasible or strict rules on literals previous in the sequence. García and Simari distinguish two different types of derivations, defeasible derivations and strict derivations. The difference between the two is that on the latter the restriction is imposed that it contains only literals in the sequence which are facts or are obtained by using a strict rule. Defeasible derivations and strict derivations are defined respectively in definition 3.5 and definition 3.6.

Derivations are monotonic; defeasibility arises from the way arguments are constructed and from the dialectical process in which arguments get assigned a status, not from the derivation of literals themselves. In addition, García and Simari observe that if a program has no facts, then it is impossible to obtain any derivation. This property of defeasible logic programs stems from the fact that it is not possible for program rules to have an empty body.

DEFINITION 3.5 (DEFEASIBLE DERIVATION) (GARCÍA & SIMARI, 2004)

Let $\mathcal{P} = (\Pi, \Delta)$ de a de.l.p. and L a ground literal. A defeasible derivation of L from $\mathcal{P}$, denoted $\mathcal{P} \mid\!\sim L$ consists of a finite sequence $L_1, L_2, \dots, L_n = L$ of ground literals, and each literal L_i is in the sequence because:

a. L_i is a fact in Π, or
b. there exists a rule R_i in $\mathcal{P}$ (strict or defeasible) with head L_i and body $B_1, B_2, \dots, B_k$ and every literal of the body is an element L_j of the sequence appearing before L_i $(j < i)$.

If in definition 3.5 $\mathcal{P}$ contains schematic rules, then $Ground(\mathcal{P})$ is used to obtain the derivation.

DEFINITION 3.6 (STRICT DERIVATION) (GARCÍA & SIMARI, 2004)

Let $\mathcal{P}$ be a de.l.p. and h a literal with a defeasible derivation $L_1, L_2, \dots, L_n = h$. We will say that h has a strict derivation from $\mathcal{P}$, denoted $\mathcal{P} \vdash h$[3], if either h is a fact or all the rules used for obtaining the sequence $L_1, L_2, \dots, L_n$ are strict rules.

In the following examples, the derivations of some of the literals in example 3.1 are shown.

EXAMPLE 3.2 (ADAPTED FROM (GARCÍA & SIMARI, 2004))

Consider de.l.p. $\mathcal{P}_{3.1}$ from example 3.1.

A derivation for literal $flies(tina)$ is the sequence $chicken(tina), scared(tina), flies(tina)$ obtained using the set of (ground-)rules $\{flies(tina) \prec chicken(tina), scared(tina)\}$. This derivation is not a strict derivation, since it is not the case that only strict rules were used.

There are multiple derivations possible for the literal $flies(tina)$. The sequence

[3] Here a deviation from the original definition in (García & Simari, 2004) was made. This formula originally was $\mathcal{P} \vdash L$, it was however assumed that this is a mistake and the formula has been replaced by $\mathcal{P} \vdash h$.

$chicken(tina), bird(tina), flies(tina)$ is also a derivation for the literal $flies(tina)$ but now obtained by using rules $\{bird(tina) \leftarrow chicken(tina), flies(tina) \prec bird(tina)\}$. Since defeasible rules were used in this derivation, this derivation is also not a strict one.

EXAMPLE 3.3 (ADAPTED FROM (GARCÍA & SIMARI, 2004))

Consider de.l.p. $\mathcal{P}_{3.1}$ from example 3.1 once more.

The derivation for the literal $flies(tweety)$ using the set of (ground-)rules $\{bird(tweety) \leftarrow penguin(tweety), flies(tweety) \prec bird(tweety)\}$ is the sequence $penguin(tweety), bird(tweety), flies(tweety)$. This derivation is clearly not strict.

It is as well possible to derive the negation of $flies(tweety)$, which is the sequence $penguin(tweety), \sim flies(tweety)$. The set of rules used in this derivation is $\{\sim flies(tweety) \leftarrow penguin(tweety)\}$. All literals in the sequence are facts or are obtained using strict rules. This derivation for $\sim flies(tweety)$ is thus a strict derivation.

To be able to define the notion of an argument, the notion of a contradictory set has to be defined first. As shown in example 3.2 and example 3.3 it is possible to have multiple derivations for a literal and moreover it is possible that there is a derivation for a literal as well as for its negation. García and Simari define contradictory sets of rules using the notion of complements, denoted by the symbol $\bar{}$ for which it holds that $\bar{p} = \sim p$ and $\overline{\sim p} = p$ (with p being a literal). In other words, a literal and its negation are complements.

DEFINITION 3.7 (CONTRADICTORY SET OF RULES) (GARCÍA & SIMARI, 2004)

A set of rules is contradictory if and only if, there exists a defeasible derivation for a pair of complementary literals from this set.

Using definition 3.7 it can be clearly seen that the set of rules $\Pi \cup \Delta$ in example 3.1 is contradictory since using this set, as there is a derivation for both $flies(tina)$ and its complement $\sim flies(tina)$. The set Π of $\mathcal{P}_{3.1}$ is not contradictory since using only rules from this set there is no derivation for two complementary literals.

In DeLP there is a convention that for a de.l.p. $\mathcal{P}$, the set Π is not contradictory. To be able to get an intuition for why this is, one has to look at how contradictions are handled in argumentation logic in general. In argumentation logic, when a contradiction arises from a set of rules, one of the complementary literals responsible for the contradiction should be defeated. In this light the convention makes sense. Π is the set of strict rules and facts and both facts and literals with a strict derivation cannot be defeated. The solution described above is thus not applicable to derivations using only elements from Π and therefore it is better to make this part of the logic non-contradictory. Literals with a defeasible derivation can be defeated, therefore if defeasible rules are also in a minimal set of rules necessary to derive two complementary literals, the set is allowed to be contradictory. Of course the subset of strict rules and facts of this set still cannot be contradictory.

At this moment the formalism of a defeasible argument structure in DeLP can be introduced. Essentially an argument structure supports a certain conclusion or a certain answer to a query. Argument structures consist of a set of defeasible rules and a conclusion. The conclusion is a literal and the set of defeasible rules includes only the rules that are used to make a derivation for this conclusion. Though the set of all defeasible and strict rules and facts of a de.l.p. may be contradictory, the union of the set of defeasible rules of an argument structure and the set of strict rules and facts are not allowed to be in contradiction. In addition, the set of defeasible rules should be minimal to derive the conclusion, in other words, there should not exist a proper subset of the set of defeasible rules, with which the conclusion also may be derived. This means that different arguments structures with the same conclusion are possible, as long as the set of defeasible rules of one of the structures is not a subset of the set of defeasible rules of the other. In the derivation of a conclusion strict rules may also be used, but these are not part of the argument structure.

DEFINITION 3.8 (ARGUMENT STRUCTURE) (GARCÍA & SIMARI, 2004)

Let h be a literal, and $\mathcal{P} = (\Pi, \Delta)$ a de.l.p. We say that $\langle \mathcal{A}, h \rangle$ is an argument structure for h, if $\mathcal{A}$ is a set of defeasible rules of Δ, such that:

1. There exists a defeasible derivation h from $\Pi \cup \mathcal{A}$
2. The set $\Pi \cup \mathcal{A}$ is non-contradictory, and
3. $\mathcal{A}$ is minimal: there is no proper subset $\mathcal{A}'$ of $\mathcal{A}$ such that $\mathcal{A}'$ satisfies both conditions 1. and 2..

The following example is based on the defeasible logic program in example 3.1. It shows some argument structures supporting some derivations in example 3.2.

EXAMPLE 3.4 (ADAPTED FROM (GARCÍA & SIMARI, 2004))

Consider de.l.p. $\mathcal{P}_{3.1}$ from example 3.1.

In example 3.2 it was shown that there are two possible derivations for the literal $flies(tina)$, it has two supporting argument structures:

- $\langle \{flies(tina) \prec bird(tina)\}, flies(tina) \rangle$

- $\left\langle \begin{matrix} \{flies(tina) \prec chicken(tina), scared(tina)\} \\ flies(tina) \end{matrix} \right\rangle$

  It can be observed that the conclusions of the argument structures are exactly the same as the literal they support. The set of defeasible rules are the same as the sets of rules used to make the derivation in example 3.2, minus the strict rules. It can also be observed that these are the grounded versions of the rules.

The constraint that the set of defeasible rules in an argument structure should be minimal does not imply that the first argument for $flies(tina)$ in example 3.4 is a proper argument and the second one is not since the first one contains less defeasible rules. The constraint actually states that there should be no subset of the set of defeasible rules which can be used to derive the conclusion. Since the set of defeasible rules in the first argument structure is not a proper subset of the set of defeasible rules in the second one, they are different arguments supporting the same conclusion.

In example 3.5 it is shown that in some cases, while a derivation of a literal is possible, there is no argument supporting it. In example 3.5 this stems from the fact that the complement of the literal may be derived using only strict rules and thus the union of the defeasible rules in the argument structure with the strict rules is a contradictory set. From this it may also be observed that while derivations are built monotonically, this is not the case for arguments, since adding strict rules or facts may invalidate arguments.

EXAMPLE 3.5 (ADAPTED FROM (GARCÍA & SIMARI, 2004))

Consider de.l.p. $\mathcal{P}_{3.1}$ from example 3.1.

In example 3.3 it was shown that there is a derivation for the literal $flies(tweety)$ as well as for its negation, however, there is only an argument structure for $\sim flies(tweety)$ and not for $flies(tweety)$. It is impossible to construct an argument for $flies(tweety)$ because of the constraint that the union of the set of defeasible rules in the argument structure with the set of strict rules and facts should be non-contradictory. $\sim flies(tweety)$ can be derived using only strict rules, so clearly such a contradiction exists for any argument with conclusion $flies(tweety)$.

The argument structure for $\sim flies(tweety)$ is $\langle \emptyset, \sim flies(tweety) \rangle$. The set of defeasible rules of this argument structure is empty, since in the derivation of this literal only strict rules were used and strict rules are not part of the argument structure.

It is not always the case that only one of two complementary literals has an argument structure, only when one of them only uses strict rules and facts. Another reason why a literal may have a derivation but not an argument structure is in the case there is another derivation of the same literal using only strict rules and facts. In that case the set of defeasible rules of the argument structure with the strict derivation is an empty set and the empty set is a subset of any set. Hence, the set of defeasible rules is not minimal for all other arguments with the same conclusion (see condition 3 of definition 3.8). García and Simari therefore conclude that if there is a strict derivation for a literal q, the argument structure for q, namely $\langle \emptyset, q \rangle$ is unique.

García and Simari define subargument structures of an argument structure to be an argument structure with a set of defeasible rules that is a subset of its super argument structure. By constraint 3 of definition 3.8, the conclusion of the subargument structure is different from the conclusion of the super argument structure (or else it is exactly the same argument structure).

DEFINITION 3.9 (SUBARGUMENT STRUCTURE) (GARCÍA & SIMARI, 2004)

An argument structure $\langle \mathcal{B}, q \rangle$ is a subargument structure of $\langle \mathcal{A}, h \rangle$ if $\mathcal{B} \subseteq \mathcal{A}$.

To recap, in example 3.6 an example for each of the notions discussed in this section.

EXAMPLE 3.6 (ADAPTED FROM (GARCÍA & SIMARI, 2004))

Consider de.l.p. $\mathcal{P}_{3.6} = \Pi \cup \Delta$ such that[4]:

$$\Pi = \begin{Bmatrix} (a \leftarrow b), (f \leftarrow g), ({\sim}h \leftarrow j), (j \leftarrow c), (k \leftarrow a, l), \\ (m \leftarrow a), ({\sim}m \leftarrow l), c, e, n, i \end{Bmatrix} \text{ and}$$

$$\Delta = \begin{Bmatrix} b \prec c, a \prec d, d \prec e, \\ {\sim}a \prec f, g \prec c, h \prec i, l \prec n \end{Bmatrix}$$

There are multiple derivations possible for the literal a, namely c, b, a and e, d, a. Based on this, two arguments structures supporting a can be constructed, namely $\langle \{b \prec c\}, a \rangle$ and $\langle \{a \prec d, d \prec e\}, a \rangle$. However, the argument structure $\langle \{b \prec c\} \cup \{a \prec d, d \prec e\}, a \rangle$ is invalid because the set of defeasible rules is not minimal. It is in addition possible to make an argument structure for the complement of a: $\langle \{{\sim}a \prec f, g \prec c\}, {\sim}a \rangle$.

The derivation for h is i, h. The derivation for the complement of h is $c, j, {\sim}h$ and is strict. The argument structure supporting the complement of h is $\langle \emptyset, {\sim}h \rangle$. There is no argument structure

[4] Brackets were added to enhance readability.

supporting h, since only strict rules are used in the derivation of $\sim h$.

There is an argument structure supporting a (see above) and there is an argument structure supporting l, namely $\langle\{l \prec n\}, l\rangle$. There is however no argument structure supporting k, such that $\langle\{a \prec d, d \prec e\} \cup \{l \prec n\}, k\rangle$, since using the set $\{a \prec d, d \prec e\} \cup \{l \prec n\} \cup \Pi$[5] derivations for both m and $\sim m$ can be constructed.

Consider argument structure $\langle\{a \prec d, d \prec e\}, a\rangle$ again. One of its subarguments is $\langle\{d \prec e\}, d\rangle$ since it is an argument structure and $\{d \prec e\} \subseteq \{a \prec d, d \prec e\}$.

The graphical representation of argument structures García and Simari use is that of a triangle with the conclusion of the argument structure at the top and the set of defeasible rules of the argument structure as the triangle itself. An example of a graphical representation of an argument structure is shown in figure 1[6].

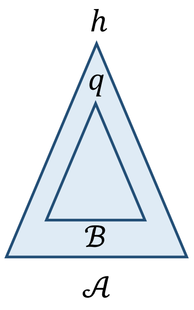


FIGURE 1: AN ARGUMENT $\langle\mathcal{A}, h\rangle$ AND A SUBARGUMENT $\langle\mathcal{B}, q\rangle$ (GARCÍA & SIMARI, 2004)

3.3 ATTACK

In DeLP it is possible for two complementary literals to have a supporting argument structure. It is intended that if this happens the 'weaker' argument gets defeated by the 'stronger' one. Methods for deciding which argument structure is stronger and establishing a preference criterion for this will be discussed in later sections. First it has to be discussed what the notions of attack and disagreement between argument structures entail.

[5] This is obviously also the case if the set of defeasible rules from the other argument supporting a is used.

[6] From this picture it is not immediately clear that the premises of $\mathcal{B}$ are a subset of the premises of $\mathcal{A}$, this is however the case.

Two literals disagree if their union with the set of strict rules and facts is contradictory, i.e. if from this set two complementary literals can be derived.

DEFINITION 3.10 (DISAGREEMENT) (GARCÍA & SIMARI, 2004)

Let $\mathcal{P} = \Pi \cup \Delta$ be a de.l.p.. We say that two literals h and h_1 disagree, if and only if the set $\Pi \cup \{h, h_1\}$ is contradictory.

If an argument structure has a conclusion which disagrees with the conclusion of a subargument structure of another argument structure, it is called the counterargument of the latter. The literal with which the counterargument disagrees is called the counterargument point and the subargument with it in its conclusion is called the disagreement subargument. It is also said that the counterargument counterargues, rebuts or attacks the super argument of the disagreement subargument.

Counterarguing can be symmetrical or non-symmetrical or respectively direct and indirect as García and Simari call it. If the counterargument point is the same as the conclusion of the counterargued argument, then the relation is symmetrical, otherwise it is not.

DEFINITION 3.11 (COUNTERARGUMENT) (GARCÍA & SIMARI, 2004)

We say that $\langle \mathcal{A}_1, h_1 \rangle$ counterargues, rebuts or attacks $\langle \mathcal{A}_2, h_2 \rangle$ at literal h, if and only if there exists a subargument $\langle \mathcal{A}, h \rangle$ of $\langle \mathcal{A}_2, h_2 \rangle$ such that h and h_1 disagree.

Below is an example of some counterarguments.

EXAMPLE 3.7 (ADAPTED FROM (GARCÍA & SIMARI, 2004))

Consider de.l.p. $\mathcal{P}_{3.1}$ of example 3.1.

$\langle \{\sim flies(tina) \prec chicken(tina)\}, \sim flies(tina) \rangle$
is a counterargument of
$\langle \{flies(tina) \prec bird(tina)\}, flies(tina) \rangle$ and vice versa since the set $\Pi \cup \sim flies(tina) \cup flies(tina)$ is clearly contradictory and thus $\sim flies(tina)$ and $flies(tina)$ disagree.
Counterarguing is symmetrical or direct in this case because the disagreement subargument is the argument structure itself.

$\langle \{\sim flies(tina) \prec chicken(tina)\}, \sim flies(tina) \rangle$ is a counterargument for $\langle \mathcal{A}, nests\_in\_trees(tina) \rangle$ with $\mathcal{A} = \left\{ \begin{matrix} nests\_in\_trees(tina) \prec flies(tina) \\ flies(tina) \prec bird(tina) \end{matrix} \right\}$, since it disagrees with the conclusion of its subargument
$\langle \{flies(tina) \prec bird(tina)\}, flies(tina) \rangle$. However in this

case the relation is not symmetrical, since the disagreement subargument is not the argument itself.

The notions of a direct and indirect counterargument are graphically represented in figure 2.

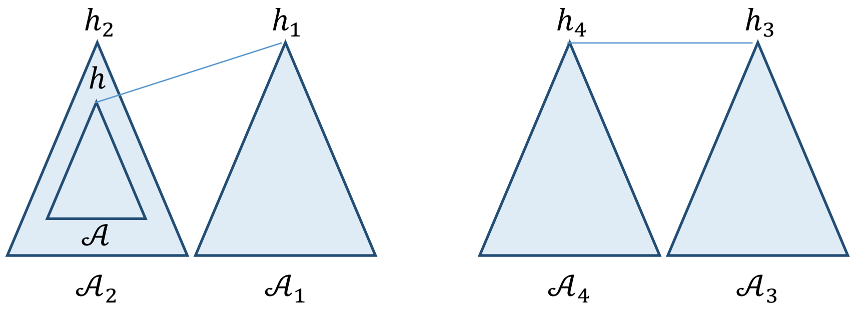


FIGURE 2: INDIRECT ATTACK (LEFT) AND DIRECT ATTACK (RIGHT) (GARCÍA & SIMARI, 2004)

It is possible that a counterargument does not have a complementary conclusion at all. In that case, disagreement arises from the union with Π. The example that García and Simari use to show this is below.

EXAMPLE 3.8 (ADAPTED FROM (GARCÍA & SIMARI, 2004))

Consider de.l.p. $\mathcal{P} = (\Pi, \Delta)$ with $\Pi = \{(h \leftarrow a), b, d, (\sim h \leftarrow c)\}$ and $\Delta = \{a \prec b, c \prec d\}$. From this program $\langle\{a \prec b\}, a\rangle$ is a counterargument for $\langle\{c \prec d\}, c\rangle$ because literals a and c disagree.

García and Simari prove that there exists no possible counterargument for an argument structure $\langle\emptyset, q\rangle$. Furthermore, they prove that such an argument structure $\langle\emptyset, q\rangle$ cannot be a counterargument for any argument structure $\langle\mathcal{A}, h\rangle$. The purpose of investigating DeLP in this thesis is mainly to be able to combine it with temporal logic and use it in a medical diagnosis system. Therefore it is just noted that these are properties of the logic and the proof is omitted. The interested reader can however find it in (García & Simari, 2004).

3.4 DEFEAT

To determine if an argument is defeated by its counterargument some kind of criterion has to be established for comparing them to each other. García and Simari describe two criteria to do this. In this section, the criteria and their incorporation in DeLP are described.

The first criterion García and Simari describe is generalized specificity. When comparing arguments using generalized specificity, arguments with greater information content, i.e. more precise arguments, are favored over arguments with less information content. In addition, arguments with less use of rules, i.e. more concise arguments, are favored of arguments which use more rules.

DEFINITION 3.12 (SPECIFICITY) (GARCÍA & SIMARI, 2004)

Let $\mathcal{P} = (\Pi, \Delta)$ be a de.l.p., and let Π_G be the set of all strict rules from Π (without including facts). Let $\mathcal{F}$ be the set of all literals that have a defeasible derivation from $\mathcal{P}$ ($\mathcal{F}$ will be considered as a set of facts). Let $\langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_2, h_2 \rangle$ be two argument structures obtained from $\mathcal{P}$. $\langle \mathcal{A}_1, h_1 \rangle$ is strictly more specific than $\langle \mathcal{A}_2, h_2 \rangle$ (denoted $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}_2, h_2 \rangle$) if the following conditions hold:

1. For all $H \subseteq \mathcal{F}$: if $\Pi_G \cup H \cup \mathcal{A}_1 \mid\!\sim h_1$ and $\Pi_G \cup H \nvdash h_1$, then $\Pi_G \cup H \cup \mathcal{A}_2 \mid\!\sim h_2$
2. There exists $H' \subseteq \mathcal{F}$ such that $\Pi_G \cup H' \cup \mathcal{A}_2 \mid\!\sim h_2$ and $\Pi_G \cup H' \cup \mathcal{A}_1 \not\mid\!\sim h_1$.

It is impossible to derive a literal using only a set of strict rules (Π_G) and a set of defeasible rules ($\mathcal{A}$). If H is a set of facts and a union of Π_G, $\mathcal{A}$ and H may make it possible to derive a literal h, H is called an activation set of $\langle \mathcal{A}, h \rangle$.

The first condition essentially states that if $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}_2, h_2 \rangle$, then if H is an activation set of an argument $\langle \mathcal{A}_1, h_1 \rangle$ and no strict derivation for h_1 can be made using H as the set of facts, then H is an activation set of $\langle \mathcal{A}_2, h_2 \rangle$. The second condition states that if $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}_2, h_2 \rangle$, then there is an activation set H' of $\langle \mathcal{A}_2, h_2 \rangle$. It in addition states that if no strict derivation for h_2 can be made using H' as the set of facts, then H' is not an activation set of $\langle \mathcal{A}_1, h_1 \rangle$.

Below is an example of arguments of which some are more specific than other ones.

EXAMPLE 3.9 (ADAPTED FROM (GARCÍA & SIMARI, 2004))

Consider de.l.p. $\mathcal{P}_{3.1}$ from example 3.1.
Assume the following:

- $\langle \mathcal{A}_1, h_1 \rangle = \langle \{\sim flies(tina) \prec chicken(tina)\}, \sim flies(tina) \rangle$,
- $\langle \mathcal{A}_2, h_2 \rangle = \langle \{flies(tina) \prec bird(tina)\}, flies(tina) \rangle$, and
- $\langle \mathcal{A}_3, h_3 \rangle = \langle \{flies(tina) \prec chicken(tina), scared(tina)\}, flies(tina) \rangle$

$\langle \mathcal{A}_1, h_1 \rangle$ is strictly more specific than $\langle \mathcal{A}_2, h_2 \rangle$ because the first is more direct than the latter. It is observable that every activation set for $\langle \mathcal{A}_1, h_1 \rangle$ is also an activation set for $\langle \mathcal{A}_2, h_2 \rangle$. An example of such an activation set is $\{chicken(tina)\}$. There is also an activation set for $\langle \mathcal{A}_2, h_2 \rangle$ which is not an activation set for $\langle \mathcal{A}_1, h_1 \rangle$, namely $\{bird(tina)\}$.

$\langle \mathcal{A}_3, h_3 \rangle$ is strictly more specific than $\langle \mathcal{A}_1, h_1 \rangle$ because it is based on more information. It can be observed that every activation set of $\langle \mathcal{A}_3, h_3 \rangle$ also activates $\langle \mathcal{A}_1, h_1 \rangle$, since every activation set activating $\langle \mathcal{A}_3, h_3 \rangle$ should contain the literals $chicken(tina)$ and $scared(tina)$. The set $\{chicken(tina)\}$ however activates $\langle \mathcal{A}_1, h_1 \rangle$, but not $\langle \mathcal{A}_3, h_3 \rangle$.

García and Simari use the definition below to define the notion of equi-specificity.

DEFINITION 3.13 (EQUI-SPECIFICITY) (GARCÍA & SIMARI, 2004)

Two arguments $\langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_2, h_2 \rangle$ are equi-specific, denoted $\langle \mathcal{A}_1, h_1 \rangle \equiv \langle \mathcal{A}_2, h_2 \rangle$, iff $\mathcal{A}_1 = \mathcal{A}_2$, and the literal h_2 has a strict derivation from $\Pi \cup \{h_1\}$, and the literal h_1 has a strict derivation from $\Pi \cup \{h_2\}$.

The other criterion described by García and Simari is on basis of explicit priorities on rules. Contrary to many other argumentation logics, rules are not compared during the inference process, but the set of defeasible rules in argument structures are compared to each other. Only priorities for defeasible rules can be given since strict rules are not defeasible and thus need not to be compared. Strict derivations however will always be preferred over defeasible ones.

DEFINITION 3.14 (PRIORITY ON RULES) (GARCÍA & SIMARI, 2004)

Let $\mathcal{P}$ de a de.l.p. and "$>$" a preference relation explicitly defined among defeasible rules. Given two argument structures $\langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_2, h_2 \rangle$, argument $\langle \mathcal{A}_1, h_1 \rangle$ will be preferred over $\langle \mathcal{A}_2, h_2 \rangle$ if:

1. There exists at least one rule $r_a \in \mathcal{A}_1$, and one rule $r_b \in \mathcal{A}_2$, such that $r_a > r_b$,
2. And there is no $r_b' \in \mathcal{A}_2$ and $r_a' \in \mathcal{A}_1$, such that $r_b' > r_a'$.

Below is an example in which two arguments are compared using the priority criterion.

EXAMPLE 3.10 (ADAPTED FROM (GARCÍA & SIMARI, 2004))

Consider de.l.p. $\mathcal{P}_{3.10} = \{(a \prec b), (\sim a \prec c), (b \leftarrow d), c, d\}$. Suppose there is a priority on the defeasible rules, such that $(a \prec b) > (\sim a \prec c)$.

Now two argument structures can be made, one supporting a, namely $\langle \{a \prec b\}, a \rangle$ and one supporting its complement $\langle \{\sim a \prec c\}, \sim a \rangle$. In this case former argument is preferred over the latter since all rules in its set with defeasible rules have a higher priority.

García and Simari mention the option of combining different comparison criteria or defining other comparison criteria than those they propose. In section 4 the choice of the comparison criterion used in the argumentation logic with temporal arguments is discussed.

Independently from some given comparison criterion denoted by $\succ$, García and Simari distinguish two different forms of defeat. The first form of defeat, called proper defeat, occurs when a counterargument $\langle \mathcal{A}_2, h_2 \rangle$ of an argument $\langle \mathcal{A}_1, h_1 \rangle$ is better according to the comparison criterion. The second form of defeat, blocking defeat, occurs when a counterargument $\langle \mathcal{A}_2, h_2 \rangle$ of an argument $\langle \mathcal{A}_1, h_1 \rangle$ is neither better nor worse according to the comparison criterion. The definitions of the two forms of defeat are below.

DEFINITION 3.15 (PROPER DEFEATER) (GARCÍA & SIMARI, 2004)

Let $\langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_2, h_2 \rangle$ be two argument structures. $\langle \mathcal{A}_1, h_1 \rangle$ is a proper defeater for $\langle \mathcal{A}_2, h_2 \rangle$ at literal h, if and only if there exists a subargument $\langle \mathcal{A}, h \rangle$ of $\langle \mathcal{A}_2, h_2 \rangle$ such that $\langle \mathcal{A}_1, h_1 \rangle$ counterargues $\langle \mathcal{A}_2, h_2 \rangle$ at h and $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}, h \rangle$.

DEFINITION 3.16 (BLOCKING DEFEATER) (GARCÍA & SIMARI, 2004)

Let $\langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_2, h_2 \rangle$ be two argument structures. $\langle \mathcal{A}_1, h_1 \rangle$ is a blocking defeater for $\langle \mathcal{A}_2, h_2 \rangle$ at literal h, if and only if there exists a subargument $\langle \mathcal{A}, h \rangle$ of $\langle \mathcal{A}_2, h_2 \rangle$ such that $\langle \mathcal{A}_1, h_1 \rangle$ counterargues $\langle \mathcal{A}_2, h_2 \rangle$ at h and $\langle \mathcal{A}_1, h_1 \rangle$ is unrelated by the preference order to $\langle \mathcal{A}, h \rangle$, i.e. $\langle \mathcal{A}_1, h_1 \rangle \nsucc \langle \mathcal{A}, h \rangle$, and $\langle \mathcal{A}, h \rangle \nsucc \langle \mathcal{A}_1, h_1 \rangle$.

Below is an example of a proper defeater and a blocking defeater.

EXAMPLE 3.11 (PROPER DEFEATER AND BLOCKING DEFEATER)

Consider arguments $\langle \mathcal{A}_1, h_1 \rangle$, $\langle \mathcal{A}_2, h_2 \rangle$, $\langle \mathcal{A}_3, h_3 \rangle$ and $\langle \mathcal{A}_4, h_4 \rangle$ such that $\langle \mathcal{A}_2, h_2 \rangle$, $\langle \mathcal{A}_3, h_3 \rangle$ and $\langle \mathcal{A}_4, h_4 \rangle$ are all counterarguments of $\langle \mathcal{A}_1, h_1 \rangle$. Consider the comparison criterion $\langle \mathcal{A}_2, h_2 \rangle \succ \langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}_4, h_4 \rangle$.

Since $\langle \mathcal{A}_2, h_2 \rangle$ is preferred over $\langle \mathcal{A}_1, h_1 \rangle$, $\langle \mathcal{A}_2, h_2 \rangle$ is a proper defeater of $\langle \mathcal{A}_1, h_1 \rangle$. Since $\langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_3, h_3 \rangle$ are unrelated according to the comparison criterion, $\langle \mathcal{A}_3, h_3 \rangle$ is a blocking defeater of $\langle \mathcal{A}_1, h_1 \rangle$. $\langle \mathcal{A}_1, h_1 \rangle$ is preferred over $\langle \mathcal{A}_4, h_4 \rangle$ and therefore $\langle \mathcal{A}_4, h_4 \rangle$ is not a defeater of $\langle \mathcal{A}_1, h_1 \rangle$.

García and Simari define an argument to be a defeater for another argument, if it is its proper defeater or its blocking defeater.

DEFINITION 3.17 (DEFEATER) (GARCÍA & SIMARI, 2004)

The argument structure $\langle \mathcal{A}_1, h_1 \rangle$ is a defeater for $\langle \mathcal{A}_2, h_2 \rangle$, if and only if either:

1. $\langle \mathcal{A}_1, h_1 \rangle$ is a proper defeater for $\langle \mathcal{A}_2, h_2 \rangle$; or
2. $\langle \mathcal{A}_1, h_1 \rangle$ is a blocking defeater for $\langle \mathcal{A}_2, h_2 \rangle$.

3.5 STATUS ASSIGNMENTS

Counterarguments are not the only argument structures that influence the defeat status of an argument structure. The ultimate status of an argument structure depends as well on counterarguments of its counterarguments and counterarguments of those arguments and so on. To determine the ultimate status of an argument structure, these also have to be evaluated.

To be able to determine the ultimate status of an argument structure, a sequence of argument structures is made starting with the argument structure for which the status needs to be determined and in which each element defeats its predecessor. Such a sequence is called an argumentation line.

DEFINITION 3.18 (ARGUMENTATION LINE) (GARCÍA & SIMARI, 2004)

Let $\mathcal{P}$ be a de.l.p. and $\langle \mathcal{A}_0, h_0 \rangle$ an argument structure obtained from $\mathcal{P}$. An argumentation line for $\langle \mathcal{A}_0, h_0 \rangle$ is a sequence of argument structures from $\mathcal{P}$, denoted $\Lambda = [\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \langle \mathcal{A}_2, h_2 \rangle, \dots]$, where each element of the sequence $\langle \mathcal{A}_i, h_i \rangle$, $i > 0$, is a defeater of its predecessor $\langle \mathcal{A}_{i-1}, h_{i-1} \rangle$.

Figure 3 depicts an argumentation line.

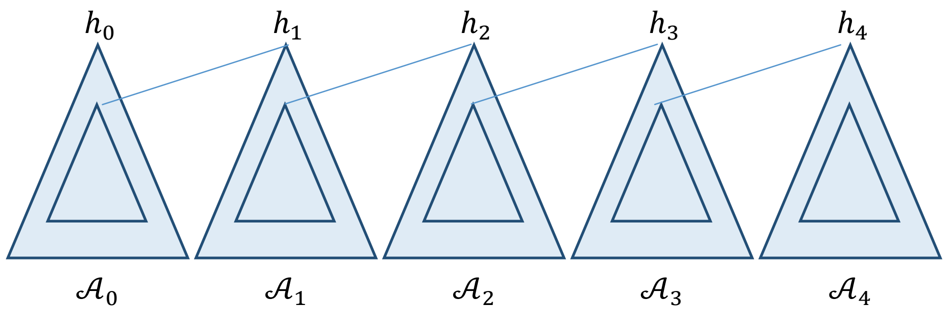


FIGURE 3: ARGUMENTATION LINE (GARCÍA & SIMARI, 2004)

Each argument structure in an argumentation line supports or interferes with the conclusion of the first argument structure in the sequence. The first argument structure obviously supports its conclusion. The second argument structure however defeats this argument and thus interferes with it. The third argument structure defeats the interfering second argument structure and is thus a supporting argument. Identifying in this way each argument structure as supporting or interfering with the conclusion of the first argument structure, two sets can be made; a set of supporting argument structures and a set of interfering argument structures.

DEFINITION 3.19 (SUPPORTING AND INTERFERING ARGUMENT STRUCTURES) (GARCÍA & SIMARI, 2004)

Let $\Lambda = [\langle\mathcal{A}_0, h_0\rangle, \langle\mathcal{A}_1, h_1\rangle, \langle\mathcal{A}_2, h_2\rangle, \dots]$ an argumentation line, we define the set of supporting argument structures $\Lambda_S = \{\langle\mathcal{A}_0, h_0\rangle, \langle\mathcal{A}_2, h_2\rangle, \langle\mathcal{A}_4, h_4\rangle, \dots\}$, and the set of interfering argument structures $\Lambda_I = \{\langle\mathcal{A}_1, h_1\rangle, \langle\mathcal{A}_3, h_3\rangle, \dots\}$.

Below is an example of an argumentation line.

EXAMPLE 3.12 (ARGUMENTATION LINES)

Consider de.l.p. $\mathcal{P}_{3.12} = \left\{\begin{matrix}(a \prec b), (\sim a \prec c), (c \prec d), \\ (\sim c \prec e), b, d, e\end{matrix}\right\}$. From this de.l.p. argument structures $\langle\mathcal{A}_0, a\rangle$, $\langle\mathcal{A}_1, \sim a\rangle$ and $\langle\mathcal{A}_2, \sim c\rangle$ can be obtained with:

- $\mathcal{A}_0 = \{(a \prec b)\}$,
- $\mathcal{A}_1 = \{(\sim a \prec c), (c \prec d)\}$, and
- $\mathcal{A}_2 = \{(\sim c \prec e)\}$.

Suppose there is an preference relation on these argument structures, such that $\langle\mathcal{A}_1, \sim a\rangle \succ \langle\mathcal{A}_0, a\rangle$, and $\langle\mathcal{A}_2, \sim c\rangle \succ \langle\mathcal{A}_1, \sim a\rangle$. An argumentation line Λ for argument structure $\langle\mathcal{A}_0, a\rangle$ is $[\langle\mathcal{A}_0, a\rangle, \langle\mathcal{A}_1, \sim a\rangle, \langle\mathcal{A}_2, \sim c\rangle]$, with the set of supporting argument structures $\Lambda_S = \{\langle\mathcal{A}_0, a\rangle, \langle\mathcal{A}_2, \sim c\rangle\}$ and the set of interfering argument structures $\Lambda_I = \{\langle\mathcal{A}_1, \sim a\rangle\}$. Figure 4 shows a graphical representation of this argumentation line.

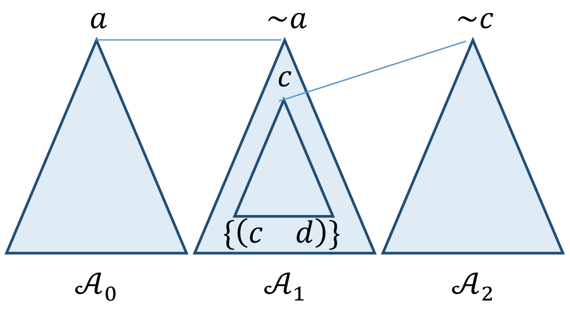


FIGURE 4: A GRAPHICAL REPRESENTATION OF THE ARGUMENTATION LINE IN EXAMPLE 3.12

There are several situations in which argumentation logics in general may exhibition unwanted behavior. In this section they are discussed briefly and superficially, since the purpose of this part of the thesis is mainly to explain the workings and structure of DeLP and not to evaluate it thoroughly. In later sections DeLP with temporal arguments will be discussed in more detail and so will be its advantages and disadvantages. I would like to refer the interested reader to (Prakken & Vreeswijk, 1998) or (García & Simari, 2004) respectively for a more in-depth analysis of problems and solutions for argumentation logics in general and for DeLP in specific.

The first problem that occurs commonly in argumentation logic has to do with self-defeating arguments, i.e. arguments that defeat themselves. Self-defeating arguments could cause argumentation lines to have an infinite length, by repeating itself infinitely in the sequence. Self-defeat is, contrary to many other argumentation logics, not a problem in DeLP since García and Simari define the notion of an argument in such a way that it could never defeat itself[7].

There are other problems very similar to the problem with self-defeating arguments that nonetheless may occur in DeLP. An argumentation line may for instance become infinite when there is a pair of argument structures that defeat each other. Another form of circular argumentation leading to an infinite argumentation line is when arguments are reintroduced in the sequence to defend themselves. In some cases this problem may as well occur when a subargument of a preceding argument in the argumentation line is reintroduced, it may then even happen that an argument is both supporting and interfering with itself.

Infinite argumentation lines are clearly undesirable; therefore García and Simari impose some additional restrictions on argumentation lines. These restrictions can be found in definitions 3.20 and 3.21.

A different kind of undesirable behavior may arise when a blocking defeater in the sequence is used to defeat a blocking defeater (of a preceding argument). Allowing such sequences is equivalent to accepting that in a blocking situation, two arguments supporting (interfering with) a sentence is preferred over having one interfering (supporting) argument. Restriction 4 of definition 3.21 prevents this problem.

DEFINITION 3.20 (CONCORDANCE) (GARCÍA & SIMARI, 2004)

Let $\mathcal{P} = (\Pi, \Delta)$ be a de.l.p.. Two arguments $\langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_2, h_2 \rangle$ are concordant iff the set $\Pi \cup \mathcal{A}_1 \cup \mathcal{A}_2$ is non-contradictory. More generally, a set of argument structures $\{\langle \mathcal{A}_i, h_i \rangle\}_{i=1}^{n}$ is concordant iff $\Pi \cup \bigcup_{i=1}^{n} \mathcal{A}_i$ is non-contradictory.

[7] See (García & Simari, 2004) for a proof.

DEFINITION 3.21 (ACCEPTABLE ARGUMENTATION LINE) (GARCÍA & SIMARI, 2004)

Let $\Lambda = [\langle \mathcal{A}_1, h_1 \rangle, \dots, \langle \mathcal{A}_i, h_i \rangle, \dots, \langle \mathcal{A}_n, h_n \rangle]$ be an argumentation line. Λ is an acceptable argumentation line iff:

1. Λ is a finite sequence
2. The set Λ_S, of supporting arguments is concordant, and the set Λ_I of interfering arguments is concordant.
3. No argument $\langle \mathcal{A}_k, h_k \rangle$ in Λ is a subargument of an argument $\langle \mathcal{A}_i, h_i \rangle$ appearing earlier in Λ $(i < k)$.
4. For all i, such that the argument $\langle \mathcal{A}_i, h_i \rangle$ is a blocking defeater for $\langle \mathcal{A}_{i-1}, h_{i-1} \rangle$, if $\langle \mathcal{A}_{i+1}, h_{i+1} \rangle$ exists, then $\langle \mathcal{A}_{i+1}, h_{i+1} \rangle$ is a proper defeater of $\langle \mathcal{A}_i, h_i \rangle$.

The first restriction in definition 3.21, obviously prevents argumentation lines from being infinite. The second restriction makes sure that there are no arguments supporting as well as interfering with the same sentence. Restriction 3 prevents subarguments from being reintroduced and restriction 4 makes sure that blocking defeaters may only be defeated by a proper defeater. García and Simari note that different restrictions are possible and by modifying these restrictions, the behavior of the formalism can be controlled. In section 4 the choice of restrictions on argumentation lines for the argumentation logic with temporal arguments is discussed.

EXAMPLE 3.13 (PAIRS OF DEFEATING ARGUMENTS)

Consider de.l.p. $\mathcal{P} = \left\{ \begin{matrix} (a \prec b), (b \prec c), (c \prec d), \\ (\sim b \prec e), (e \prec \sim a), (\sim a \prec f), d, f \end{matrix} \right\}$. From $\mathcal{P}$ two argument structures can be obtained, namely $\langle \mathcal{A}_0, a \rangle$ and $\langle \mathcal{A}_1, \sim b \rangle$, with:

- $\mathcal{A}_0 = \{(a \prec b), (b \prec c), (c \prec d)\}$, and
- $\mathcal{A}_1 = \{(\sim b \prec e), (e \prec \sim a), (\sim a \prec f)\}$.

Now it is possible that argument structure $\langle \mathcal{A}_0, a \rangle$ defeats $\langle \mathcal{A}_1, \sim b \rangle$, by defeating its subargument structure $\langle \{(\sim a \prec f)\}, \sim a \rangle$, while at the same time $\langle \mathcal{A}_1, \sim b \rangle$ defeats $\langle \mathcal{A}_0, a \rangle$ by defeating its subargument structure $\langle \{(b \prec c), (c \prec d)\}, b \rangle$. If there were no restrictions like imposed in definition 3.21, an infinite argumentation line for $\langle \mathcal{A}_0, a \rangle$ could be made, namely $[\langle \mathcal{A}_0, a \rangle, \langle \mathcal{A}_1, \sim b \rangle, \langle \mathcal{A}_0, a \rangle, \langle \mathcal{A}_1, \sim b \rangle, \dots]$, which would be undesirable. However, if the restrictions in definition 3.21 are taken into account, such an argumentation line would be not acceptable, since it is infinite and this is not acceptable by restriction 1 and moreover an argument is a subargument of itself and thus is not allowed to appear more than once in the argumentation line according to restriction 3.

There may be multiple defeaters for an argument and thus multiple argumentation lines. Arguments in these argumentation lines may also have multiple defeaters and this means even more argumentation lines can be generated. All these possible argumentation lines for an argument structure can be joined into a dialectical tree for this argument structure.

The root of a dialectical tree for an argument structure $\langle\mathcal{A}, h\rangle$ is labeled $\langle\mathcal{A}, h\rangle$. The children of a node are labeled with the corresponding defeaters of the argument structure in its label. If a node is a leaf, the argument structure in its label is undefeated. In a dialectical tree, argument structures represented in the nodes in each path from the root of the tree to a leaf corresponds to one of the acceptable argumentation lines for the argument structure represented by the root.

DEFINITION 3.22 (DIALECTICAL TREE) (GARCÍA & SIMARI, 2004)

Let $\langle\mathcal{A}_0, h_0\rangle$ be an argument structure from a program $\mathcal{P}$. A dialectical tree for $\langle\mathcal{A}_0, h_0\rangle$, denoted $\mathcal{T}_{\langle\mathcal{A}_0,h_0\rangle}$, is defined as follows:

1. The root of the tree is labeled with $\langle\mathcal{A}_0, h_0\rangle$.
2. Let N be a non-root node of the tree labeled $\langle\mathcal{A}_n, h_n\rangle$, and $\Lambda = [\langle\mathcal{A}_0, h_0\rangle, \langle\mathcal{A}_1, h_1\rangle, \langle\mathcal{A}_2, h_2\rangle, \dots, \langle\mathcal{A}_n, h_n\rangle]$ the sequence of labels of the path from the root to N. Let $\langle B_1, q_1\rangle, \langle B_2, q_2\rangle, \dots, \langle B_k, q_k\rangle$ be all the defeaters for $\langle\mathcal{A}_n, h_n\rangle$.
   For each defeater $\langle B_i, q_i\rangle$ $(1 \le i \le k)$, such that, the argumentation line $\Lambda' = [\langle\mathcal{A}_0, h_0\rangle, \langle\mathcal{A}_1, h_1\rangle, \langle\mathcal{A}_2, h_2\rangle, \dots, \langle\mathcal{A}_n, h_n\rangle, \langle B_i, q_i\rangle]$ is acceptable, then the node N has a child N_i labeled $\langle B_i, q_i\rangle$.
   If there is no defeater for $\langle\mathcal{A}_n, h_n\rangle$ or there is no $\langle B_i, q_i\rangle$ such that Λ' is acceptable, then N is a leaf.

Below is an example of a dialectical tree in DeLP.

EXAMPLE 3.14 (DIALECTICAL TREE) (GARCÍA & SIMARI, 2004)

Consider the following de.l.p.:

$$\left\{\begin{matrix} (a \prec b), (\sim b \prec e), (\sim b \prec c, f), (\sim f \prec i), (b \prec c), (f \prec g), \\ (\sim f \prec g, h), (\sim h \prec k), (\sim b \prec c, d), (h \prec j), e, i, c, g, k, d, j \end{matrix}\right\}$$

Here the literal a is supported by $\langle\mathcal{A}, a\rangle = \langle\{(a \prec b), (b \prec c)\}, a\rangle$ and there exist three defeaters for it, each of them starting three different argumentation lines: $\langle\mathcal{B}_1, \sim b\rangle = \langle\{(\sim b \prec c, d)\}, \sim b\rangle$,

$\langle \mathcal{B}_2, {\sim}b \rangle = \langle \{(\sim b \prec c, f), (f \prec g)\}, {\sim}b \rangle$ and $\langle \mathcal{B}_3, {\sim}b \rangle = \langle \{(\sim b \prec e)\}, {\sim}b \rangle$. The first two are proper defeaters and the last one is a blocking defeater.

Observe that the argument structure $\langle \mathcal{B}_1, {\sim}b \rangle$ has the counterargument $\langle \{(b \prec c)\}, b \rangle$, but it is not a defeater since the former is more specific.[8] Thus no defeaters for $\langle \mathcal{B}_1, {\sim}b \rangle$ exist and the argumentation line ends there.
The argument structure $\langle \mathcal{B}_3, {\sim}b \rangle$ has a blocking defeater: $\langle \{(b \prec c)\}, b \rangle$. Note that $\langle \{(b \prec c)\}, b \rangle$ is the disagreement subargument of $\langle \mathcal{A}, a \rangle$, therefore, it cannot be introduced because it produces an argumentation line that is not acceptable.

The argumentation structure $\langle \mathcal{B}_2, {\sim}b \rangle$ has two defeaters that can be introduced: $\langle \mathcal{C}_1, {\sim}f \rangle = \langle \{(\sim f \prec g, h), (h \prec j)\}, {\sim}f \rangle$ (proper defeater) and $\langle \mathcal{C}_2, {\sim}f \rangle = \langle \{(\sim f \prec i)\}, {\sim}f \rangle$ (blocking defeater).

Thus one of the lines is split in two argumentation lines. The argument $\langle \mathcal{C}_1, {\sim}f \rangle$ has a blocking defeater that can be introduced in the line: $\langle \mathcal{D}_1, {\sim}h \rangle = \langle \{(\sim h \prec k)\}, {\sim}h \rangle$.
Finally observe that both $\langle \mathcal{D}_1, {\sim}h \rangle$ and $\langle \mathcal{C}_2, {\sim}f \rangle$ have a blocking defeater, but they cannot be introduced because they make the argumentation line not acceptable.

The dialectical tree for $\langle \mathcal{A}, a \rangle$ is shown in figure 5.

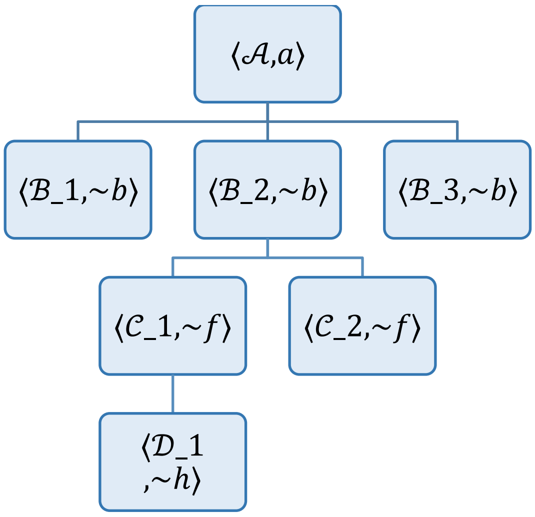


FIGURE 5: DIALECTICAL TREE FOR EXAMPLE 3.14 (GARCÍA & SIMARI, 2004)

[8] García and Simari use generalized specificity as the comparison criterion here.

García and Simari observe that a subtree of a dialectical tree is itself not always a dialectical tree since some defeaters that are not included in the super tree because they would make an argumentation line unacceptable are acceptable in the subtree.

A sentence h in DeLP is warranted or justified if there is an argument structure $\langle \mathcal{A}, h \rangle$ that is undefeated. A dialectical tree with $\langle \mathcal{A}, h \rangle$ at the root can be built to determine whether it is undefeated. Each leaf in this tree is marked undefeated (U), since it has no defeaters. Each inner node in the tree will get the mark defeated (D) if one or more of its children is undefeated and will get the mark undefeated otherwise. By marking each node in the tree, it is eventually possible to mark the root node $\langle \mathcal{A}, h \rangle$ and conclude whether it is defeated or undefeated and thus whether a sentence h is warranted or not. Below is the procedure by García and Simari for marking the nodes in a tree and the definition of warranted literals.

PROCEDURE 3.1 (MARKING OF A DIALECTICAL TREE) (GARCÍA & SIMARI, 2004)

Let $\mathcal{T}_{\langle \mathcal{A}, h \rangle}$ be a dialectical tree for $\langle \mathcal{A}, h \rangle$. The corresponding marked dialectical tree denoted $\mathcal{T}^*_{\langle \mathcal{A}, h \rangle}$, will be obtained marking every node in $\mathcal{T}_{\langle \mathcal{A}, h \rangle}$ as follows:

1. All leaves in $\mathcal{T}_{\langle \mathcal{A}, h \rangle}$ are marked as "U"'s in $\mathcal{T}^*_{\langle \mathcal{A}, h \rangle}$.
2. Let $\langle \mathcal{B}, q \rangle$ be an inner node of $\mathcal{T}_{\langle \mathcal{A}, h \rangle}$. Then $\langle \mathcal{B}, q \rangle$ will be marked as "U" in $\mathcal{T}^*_{\langle \mathcal{A}, h \rangle}$ iff every child of $\langle \mathcal{B}, q \rangle$ is marked as "D". The node $\langle \mathcal{B}, q \rangle$ will be marked as "D" in $\mathcal{T}^*_{\langle \mathcal{A}, h \rangle}$ iff it has at least a child marked as "U".

DEFINITION 3.23 (WARRANTED LITERALS) (GARCÍA & SIMARI, 2004)

Let $\langle \mathcal{A}, h \rangle$ be an argument structure and $\mathcal{T}^*_{\langle \mathcal{A}, h \rangle}$ its associated marked dialectical tree. The literal h is warranted iff the root of $\mathcal{T}^*_{\langle \mathcal{A}, h \rangle}$ is marked as "U". We will say that $\mathcal{A}$ is a warrant for h.

Below is an example of a marked dialectical tree.

EXAMPLE 3.15 (MARKED DIALECTICAL TREE) (ADAPTED FROM (GARCÍA & SIMARI, 2004))

Below is dialectical tree from figure 5, marked according to procedure 3.1.

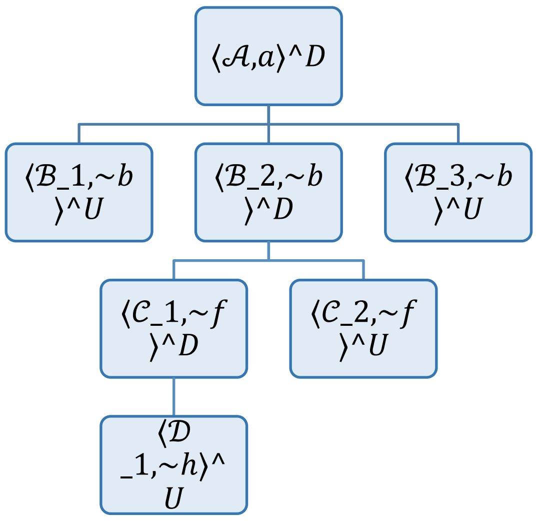


FIGURE 6: THE MARKED DIALECTICAL TREE FOR EXAMPLE 3.15 (GARCÍA & SIMARI, 2004)

From figure 6 it can be observed that literal a is not warranted.

Using the definitions above, García and Simari prove that literals that have a strict derivation are always warranted.

Building an entire dialectical tree for an argument structure and marking all nodes is clearly not the most efficient way to determine whether a literal is warranted. Consider example 3.15 again. a is only warranted if all children of $\langle \mathcal{A}, a \rangle$ are defeated (and there is no other argument structure with a at its conclusion which is ultimately undefeated). To know whether $\langle \mathcal{A}, a \rangle$ is marked defeated, it is not necessary to build the entire marked dialectical tree. As soon as it is clear that one the children of $\langle \mathcal{A}, a \rangle$ is marked undefeated, $\langle \mathcal{A}, a \rangle$ can be marked defeated and the rest of the marked dialectical tree does not have to be built anymore. In case of example 3.15, as soon as it is clear that $\langle \mathcal{B}_1, {\sim}b \rangle$ or $\langle \mathcal{B}_3, {\sim}b \rangle$ is undefeated, $\langle \mathcal{A}, a \rangle$ can be marked defeated and the rest of the dialectical tree is of no importance anymore.

García and Simari developed a more efficient procedure for determining whether a literal is warranted, pruning nodes in the dialectical tree that do not contribute to the status of the root. The procedure considers each branch of the leaf depth-first and from left to right. To decide whether a literal h is warranted, the procedure considers each argument structure for h in turn. García and Simari give no formal description for the procedure, but they use the example below to describe it.

EXAMPLE 3.16 (WARRANT PROCEDURE WITH PRUNING) (GARCÍA & SIMARI, 2004)

Suppose that in order to find a warrant for h_1, the argument $\mathcal{A}_1$ is found, and the acceptable argumentation line $[\langle \mathcal{A}_1, h_1 \rangle, \langle \mathcal{A}_2, h_2 \rangle, \langle \mathcal{A}_3, h_3 \rangle, \langle \mathcal{A}_4, h_4 \rangle, \langle \mathcal{A}_5, h_5 \rangle]$ is built. In this situation, the acceptable argumentation line ends with the supporting argument $\mathcal{A}_5$, so the marking procedure establishes that $\langle \mathcal{A}_1, h_1 \rangle$ is up to this point a U. However, the warrant process cannot finish there because there could be more defeaters to consider. Therefore, the process will continue expanding other argumentation lines.

First, note that although there could be more defeaters for $\mathcal{A}_4$, considering them will not change $\mathcal{A}_4$'s status because of $\mathcal{A}_5$. Therefore the tree can be pruned at that point without losing further defeaters for $\mathcal{A}_4$.

However, the previous analysis does not apply to $\mathcal{A}_3$, because if an undefeated defeater is found for it, the mark of $\mathcal{A}_3$ could change. It is for this reason the procedure will look for any other possible defeater $\mathcal{A}_4{}'$ for $\mathcal{A}_3$, creating a new argumentation line.

If a defeater $\mathcal{A}_4{}'$ is found (with no defeaters for it), then the argumentation line will end with an interfering argument, and therefore $\mathcal{A}_1$will be a D. Again, pruning could be influenced, because although there could be more defeaters for $\mathcal{A}_3$, they cannot modify its status. However, there might be another defeater $\mathcal{A}_3{}'$ for $\mathcal{A}_2$, creating in that case a new argumentation line.

Being able to determine whether a literal is warranted means that queries about literals can be answered. García and Simari use a modal belief-operator B such that for a literal h, Bh means that h is warranted and $\neg Bh$ means that h is not warranted.

DEFINITION 3.24 (ANSWER TO QUERIES) (GARCÍA & SIMARI, 2004)

The answer of a DeLP interpreter can be defined in terms of a modal operator B. In terms of B, there are four possible answers for a query h:

- YES, if Bh (h is warranted)
- NO if $B\overline{h}$ (the complement of h is warranted)
- $UNDECIDED$, if $\neg Bh$ and $\neg B\overline{h}$ (nether h nor $\overline{h}$ are warranted)
- $UNKNOWN$, if h is not in the language of the program.

4 THE PROPOSED TEMPORAL ARGUMENTATION LOGIC

As described in section 2.5 medical diagnostic systems are required to handle temporal information, they should furthermore be able to handle partially missing or imprecise temporal information. Even though argumentation logics are capable of dealing with facts that are missing as a whole, they are generally not equipped to handle cases in which part of the information of a fact is present and part is not. Therefore, merely expanding the argumentation logic with temporal arguments is insufficient.

In a derivation process, to infer the head of a rule, all literals in its body need to be matched with facts or literals that have already been derived. If the literals in the body contain temporal arguments, these need to be also present in the corresponding facts. Below is an example in which temporal information is missing for one of the facts in a logic with temporal arguments. To show the problem logics in general have when a part of a fact is missing, it is not necessary to specify the logic used in the example below any further, which therefore has not been done.

EXAMPLE 4.1 (MISSING TEMPORAL INFORMATION)

Consider a rule $tired(x) \leftarrow sleeping(x, t_1, t_2) \wedge (t_2 - t_1 < 8) \wedge working(x, t_3, t_4) \wedge (t_4 - t_3 > 8)$ in some logical system denoting that "if someone slept for less than 8 hours and worked for more than 8 hours, then they are tired". t_1, t_2, t_3 and t_4 are temporal arguments denoting start en ending times of the intervals at which the predicates are true.

Consider the following facts in the same system: $sleeping(mary, 1{,}6)$ and $working(mary)$. To be able to infer the head of the rule above, the body needs to be determined to be true.

$sleeping(x, t_1, t_2\ ) \wedge (t_2 - t_1 < 8)$ can be determined to be true since there is a fact $sleeping(mary, 1{,}6)$ and $6 - 1 < 8$. $working(x, t_3, t_4) \wedge (t_4 - t_3 > 8)$ can however not be determined to be true, since the fact $working(mary)$ does not contain the required temporal information.

Since there is no temporal information on how long Mary was working, we are not able to infer anything in the example above and information that Mary was working is disregarded. It is however undesirable to do this in many cases, since potentially useful information is not used. On the other hand, we cannot just add a fact saying that Mary was working for more than 8 hours, since we do not know that for certain.

In this case, it would be preferable to derive $tired(mary)$ under the assumption that Mary was working for more than 8 hours. On the other hand, if stronger information becomes available that Mary was not working more than 8 hours or that Mary is not tired, we would like $tired(mary)$ to be defeated. In the temporal argumentation logic proposed in this thesis, this is exactly the way in which partially missing temporal information is handled. The logic is designed to make optimal use of information that is available, while carefully keeping track of assumptions that need to be made when information is missing.

In example 4.1, implicitly there is some temporal information about $working(mary)$, namely that it could possibly have any duration, including a duration of more than 8 hours. This kind of knowledge could be represented by replacing the temporal arguments in predicates that denote the exact time at which they became true and the exact time at which they ceased to be true with a set of all intervals at which the argument possibly could be true according to the available information. The facts from example 4.1 would in that case be written as $sleeping(mary, \{(1,6)\})$ and $working(mary, \mathcal{I})$ where $\mathcal{I}$ denotes a set containing all possible intervals in the range of time that is considered by the system in which the logic is used. In the proposed temporal argumentation logic, sets containing intervals, such as $\{(1,6)\}$ and $\mathcal{I}$ are called interval sets.

To deal with interval sets, the rule in example 4.1 should be rewritten as well. The corresponding rule in the proposed logic is in the example below. It expresses the same information as the rule in example 4.1, but due to its different form it is possible to use interval sets and to infer on which assumptions its head can be derived. How exactly rules of this form should be read and understood will become more apparent in the course of this section.

EXAMPLE 4.2 (RULES AND FACTS IN THE PROPOSED LOGIC)

The following rules and facts are in the language of the proposed logic and they replace the rules and facts in example 4.1:

- $tired(x) \leftarrow sleeping(x, \mathcal{H}), working(x, \mathcal{K}),$
  $\mathcal{H} \times \mathcal{I} \cap \{\langle i_1, i_2 \rangle | duration(i_1) < 8 \wedge duration(i_2) > 8\}$
  $\neq \emptyset$
- $sleeping(mary, \{(1,6)\})$
- $working(mary, \mathcal{I})$

In the rules and facts above, $\mathcal{I}$, $\mathcal{H}$ and $\mathcal{K}$ are called interval sets and i_1 and i_2 are intervals. $\mathcal{I}$ contains all intervals in the time span considered by a system that uses the proposed logic. $\mathcal{H} \times \mathcal{I} \cap \{\langle i_1, i_2 \rangle | duration(i_1) < 8 \wedge duration(i_2) > 8\} \neq \emptyset$ is called a temporal constraint. A temporal constraint in the proposed logic expresses that the head of the rule may only be derived if it is possible that the intervals at which the literals in its body are true have the desired properties.

In the proposed logic, there is a defeasible derivation of $tired(mary)$. This literal is derived under the assumption that Mary worked for more than 8 hours. If information becomes available that Mary is not tired or did not work for more than 8 hours and this information is based on stronger assumptions, then $tired(mary)$ is defeated during the argumentation process in the proposed logic.

In the first subsection of this section, some ideas that are essential to understanding the proposed logic are discussed. In subsequent subsections, the proposed temporal argumentation logic is defined formally. Most definitions are inspired by the definitions from (García & Simari, 2004) or are taken from it. The definitions of (García & Simari, 2004) for the system DeLP are discussed in chapter 3.

4.1 FUNDAMENTALS OF THE PROPOSED LOGIC

Literals containing a term that denotes an interval set are called temporal literals in the proposed logic. Their semantics are crucial to understanding the proposed logic and the thoughts behind it. It is therefore important to discuss some ideas on which the proposed logic was built prior to formally defining the logic.

The information expressed in a temporal literal in the proposed logic differs from the information that is usually expressed in a literal. To establish their semantics, it is necessary to have some concept of the “real world” that is represented by the facts and rules in a logical system. An elaborate philosophical discussion of what this “real world” is, is outside the scope of this thesis. Therefore, in this thesis the “real world” is defined in a short and simplified way, while avoiding controversy to the extent possible.

The “real world” is whatever is represented by the rules and facts in a logical system, i.e. the domain of discourse. The actual situation in this “real world” is a collection of things that are true or false at certain intervals. It is assumed that the actual situation in the “real world” is such that contradictions do not exist. In other words, the actual situation in the “real world” cannot be such that there are things that are true and false at the same time. The things that are true and that are false can be represented by corresponding literals in the logic.

It will be assumed that information about the actual situation in the “real world” can be incomplete. More specific, information may be completely or partially missing about at which intervals some things are true or false. Partially missing information still gives some information about the actual situation in the “real world”, namely which situations are possibly the actual situation in the “real world”. Temporal literals thus contain information about which situations in the “real world” are considered possible. It is assumed that information that is expressed by temporal literals that are facts in the logic is correct. In other words, of all the possible situations expressed by all facts in the logic, one is a partial or complete representation of the actual situation in the “real world”.

The semantics and syntax of temporal literals are defined formally in section 4.2. The formal semantics of temporal literals are founded on the ideas described above.

4.2 ATOMIC FORMULAS AND INTERVAL SETS

The notion of an interval set is built upon the notions of an interval and a set with natural numbers that denote the points in time that are considered by the system.

DEFINITION 4.1 (INTERVAL SETS)

$\mathcal{T}$ is a set of natural numbers such that given some natural number $n \in \mathcal{T}$, $\mathcal{T} = \{i \in \mathbb{N} | 0 \leq i \leq n\}$.

An interval in the proposed logic is a pair (i^-, i^+), where:

- $i^-, i^+ \in \mathcal{T}$, and
- $i^- < i^+$

An interval set $\mathcal{I} = \{(i_1^-, i_1^+), \ldots, (i_k^-, i_k^+)\}$ is a non-empty finite set of intervals.

REMARKS

$\mathcal{T}$ is the set of all points in time considered by the system using the logic, denoted by natural numbers. The natural number $0 \in \mathcal{T}$ denotes the first point in time considered by the system and n denotes the last point in time considered by the system. For each $i, j \in \mathcal{T}$, if $i < j$, then i denotes a point in time preceding the point in time denoted by j.

For an interval (i^-, i^+), i^- denotes the point in time the interval starts and i^+ denotes the point in time the interval ends.

Set $\mathcal{I}_{all} = \{(i^-, i^+) | i^-, i^+ \in \mathcal{T}\}$ is the interval set containing all possible intervals based on $\mathcal{T}$. The set of all possible interval sets in the time considered by a system is the power set of $\mathcal{I}_{all}$, $\mathcal{P}(\mathcal{I}_{all})$.

The symbols from the set $\{\mathscr{i}, \mathscr{j}, \mathscr{k}, \ldots\}$ of lower case script letters may be used as an abbreviations to denote intervals, e.g. interval (i^-, i^+) may be denoted by $\mathscr{i}$.

From definition 4.1 it can be observed that set $\mathcal{T}$ is finite. Time is usually considered infinite. Defining the set of time points as infinite would make the set of possible intervals infinite as well, which in turn would allow for infinite interval sets. This would severely heighten the complexity of the definitions of the functions on these interval sets. While it probably is possible to define $\mathcal{T}$ as infinite, it was decided not to do so to guard the intelligibility of the proposed logic and its definitions.

Defining $\mathcal{T}$ as finite also makes sense from a more practical point of view since it would be pointless for a medical diagnostic system to take infinite time into account. When the proposed logic is used, the size and content of set $\mathcal{T}$ should match the domain in which the proposed logic is used. The timespan, i.e. the value of n may vary for different domains. For certain sets of diagnoses, it may be optimal to consider a time span of a couple of weeks and for others it may be optimal to consider the time span of a person's whole life. Correspondingly, the natural numbers in $\mathcal{T}$ may denote different units of time. In some cases, it may be optimal to take seconds as units, while in others it may be optimal to let the elements of $\mathcal{T}$ denote hours, weeks or months.

Intervals in the proposed logic are defined to be pairs of natural numbers. The first number denotes the point in time the interval starts and the second number denotes the point in time interval ends. Semantically, it is implicitly assumed that every point in time between this starting point and ending point is also in the interval. This assumption is however not made explicit since this assumption does not influence the syntactic definitions of the other notions. Intervals in addition are defined to always have a duration of 1 time unit or more, as there is no purpose for intervals with a duration of 0 if the elements in $\mathcal{T}$ are chosen correctly.

Interval sets are sets containing intervals. They are used as terms in formulas. To be able to incorporate them into the language of the logic, the notion of an atomic formula has to be redefined.

DEFINITION 4.2 (ATOMIC FORMULAS)

Given a set of variable symbols $V = \{x, y, z, \dots\}$, a set of symbols for interval sets $I = \{\mathcal{H}, \mathcal{I}, \mathcal{J}, \mathcal{K} \dots\}$ and a set of function symbols $F = \{f, g, h, c \dots\}$, terms are defined as follows:

- Any variable $x \in V$ is a term,
- Any interval set $\mathcal{I} \in I$ is a term,
- Any function $f(t_1, \dots, t_n)$, such that $f \in F$ is a function with arity n where each t_i is a term, is a term. If a function has an arity of 0, it is called a constant.

Given a set of predicate symbols $S = \{P, Q, R, \dots\}$, an expression $P(t_1, \dots, t_n)$, where $P \in S$ is an n-ary predicate symbol and $t_1, \dots, t_n$ are terms, is an atomic formula.

REMARKS

Commonly only variables and functions on terms are considered to be terms. In this definition interval sets are considered to be terms as well. Formally, this is not exactly accurate. A term is not an interval set (or a variable etc.), but can be interpreted as an interval set under an interpretation.

If D is a nonempty domain of discourse, then set $\mathcal{T}$ is the subset of D containing only elements that are time points. Which interval sets can possibly exist, depends on the elements of $\mathcal{T}$. Therefore, a symbol denoting an interval set can be interpreted as an element of the set of interval sets, $\mathcal{P}(\mathcal{I}_{all})$ (of which the content is determined by the content of set $\mathcal{T}$). Correspondingly, functions have different interpretations depending on whether they contain interval sets as terms and whether their range is $\mathcal{P}(\mathcal{I}_{all})$.

$=$ and $\neq$ are predicate symbols in S denoting equality and inequality respectively. $\cap$ and $\times$ are a function symbols in F denoting intersection and the n-ary Cartesian product respectively. $\emptyset$ is a constant symbol in F denoting the empty set. These predicates and functions all have their usual definitions. Their infix notation can be used as syntactic sugar.

$c \in F$ is usually a constant.

Below are some examples of atomic formulas. Some of them contain interval sets and some of them don't.

EXAMPLE 4.3 (ATOMIC FORMULAS)

The following formulas are atomic formulas:

- $P(c)$
- $P(x, \mathcal{I})$
- $P\left(\cap\left(\mathcal{I}, f(x)\right)\right)$
- $P\left(x, f\left(g(x)\right), h(\mathcal{I})\right)$
- $\mathcal{I} \times \mathcal{H} \cap \{\langle i_1, i_2 \rangle\} \neq \emptyset$
- $fever\left(x, during(\{(5,8), (15,19)\})\right)$

The following are not atomic formulas:

- $P(P)$
- $P(x) \land Q(x)$
- $\mathcal{I} \times \mathcal{H}$
- $\mathcal{I}$

Literals usually are defined to be atomic formulas, since the proposed logic is based on DeLP the definition of a literal of (García & Simari, 2004) was used.

DEFINITION 4.3 (FACTS OR LITERALS)

A fact or a literal $P(t_1, \ldots, t_n)$ is a ground atomic formula or its negation.

REMARKS

A ground term is a term containing no variables, hence constants, interval sets and functions on ground terms are ground terms.

A ground atomic formula is an atomic formula with only ground terms.

EXAMPLE 4.4 (FACTS OR LITERALS)

The following atomic formulas are facts or literals:

- $P(c)$
- $P(\mathcal{I}, \mathcal{I})$
- $\neg P\left(\cap\left(\mathcal{I}, f(c)\right)\right)$
- $P\left(c, f\left(g(c)\right), h(\mathcal{I})\right)$
- $\mathcal{I} \times \mathcal{H} \cap \mathcal{I} \times \mathcal{K} \neq \emptyset$
- $\neg fever\left(mary, during(\{(5,8), (15,19)\})\right)$

The following atomic formulas are not facts or literals:

- $\neg P(x, \mathcal{I})$
- $P\left(f\left(g(x)\right)\right)$
- $P(x, c)$

The word ‘fact’ is commonly used to refer to ground atomic formulas (or their negations) that are axioms in the system or serve as input to a system using the proposed logic. The word ‘literal’ is used to refer to ground atomic formulas and their negations, which are derived or are part of the body or the head of a rule. Both words refer to the same kind of formulas syntactically.

Literals may or may not contain terms denoting interval sets. It is important to distinguish literals containing one term that denotes an interval set from literals containing multiple or no such terms, as their semantics are different in the proposed logic. A term denoting an interval set or a function that returns an interval set is called a temporal term. Literals containing exactly one temporal term, are named temporal literals.

The definition of a temporal term is a bit unusual in the sense that a temporal term is defined by both its syntax as well as its semantics. To avoid this, it would be necessary to make a syntactic distinction between functions that have a subset of $\mathcal{P}(\mathcal{I}_{all})$ as their range and other functions. This would however be detrimental rather than beneficial to the intelligibility of the definitions of the proposed logic.

DEFINITION 4.4 (TEMPORAL TERMS)

A term t is a temporal term, if and only if:

- $t = \mathcal{I}$, or
- $t = f(t_1, \dots, t_n)$, where the range of f is a subset of $\mathcal{P}(\mathcal{I}_{all})$ and $t_1, \dots, t_n$ are ground terms.

Below are some examples of temporal terms.

EXAMPLE 4.5 (TEMPORAL TERMS)

The following are temporal terms:

- $\mathcal{I}$
- $f(\mathcal{I})$, where the range of f the set of interval sets
- $f(\mathcal{I}, \mathcal{H})$, where the range of f is the set of interval sets
- $g(f(\mathcal{I}), \mathcal{H})$, where the range of f and g is the set of interval sets
- $during(\{(5,8), (15,19)\})$

The following are not temporal terms:

- $f(c)$
- $f(\mathcal{I}, c)$
- c
- x

The notion of a temporal term is used to define the notion of a temporal literal. The semantics of temporal literals are discussed in section 4.1.

DEFINITION 4.5 (TEMPORAL LITERALS)

A temporal literal $P(t_1, \dots, t_n)$ or its negation is a literal that has exactly one temporal term t_i as term. Such a term t_i is said to be the temporal term of $P(t_1, \dots, t_n)$.

Below are some examples of temporal literals.

EXAMPLE 4.6 (TEMPORAL LITERALS)

The following literals are temporal literals with their temporal terms:

- $P(\mathcal{I})$, with temporal term $\mathcal{I}$
- $\neg P(c, \mathcal{I})$, with temporal term $\mathcal{I}$
- $P(c_1, f(c_2), g(\mathcal{I}), c_2)$ where the range of f is a subset of the power set of $\mathcal{I}_{all}$ and the range of g is not, with temporal term $f(c_2)$
- $P\left(f\big(g(\mathcal{I})\big)\right)$, where the range of f is a subset of the power set of $\mathcal{I}_{all}$, with temporal term $f\big(g(\mathcal{I})\big)$
- $fever\big(mary, during(\{(5{,}8), (15{,}19)\})\big)$, with temporal term $during(\{(5{,}8), (15{,}19)\})$

The following are not temporal literals:

- $P(c)$
- $= (c, c)$
- $P\big(\mathcal{I}, f(\mathcal{H})\big)$
- $\neg P(\mathcal{I}, \mathcal{I})$

The semantics of temporal literals are based on the ideas discussed in section 4.1. The information about the actual situation in the "real world" contained in a temporal literal $P(t_1, \dots, t_n)$ with its term t_j denoting an interval set $\mathcal{I}$, is that the relationship P between $t_1, \dots, t_{j-1}, t_{j+1}, \dots, t_n$ holds at exactly one interval $\imath \in \mathcal{I}$. Such a temporal literal does however not contain information stating which of the intervals in $\mathcal{I}$ is the interval at which the relationship holds in the actual situation in the "real world". Correspondingly, the information contained in a temporal literal $\neg P(t_1, \dots, t_n)$ with its term t_j denoting an interval set $\mathcal{I}$ is that the relationship P between $t_1, \dots, t_{j-1}, t_{j+1}, \dots, t_n$ does not hold at exactly one interval $\imath \in \mathcal{I}$. If $P(t_1, \dots, t_n)$ is a temporal literal with temporal term t_j that denotes an interval set $\mathcal{I}$ and we would like to address a situation in which the relation P holds between $t_1, \dots, t_{j-1}, t_{j+1}, \dots, t_n$ at interval $\imath \in \mathcal{I}$, then for the sake of brevity we say that $P(t_1, \dots, t_n)$ is true at $\imath$ in this situation.

Consider a temporal literal $P(t_1, \dots, t_n)$ with its term t_j denoting an interval set $\mathcal{I}$. Suppose $\mathcal{I}$ contains more than one element. In this case, it is only known that $P(t_1, \dots, t_n)$ is true at one of the intervals in $\mathcal{I}$ in the actual situation, but it is not known exactly at which one. This means that information about the situation in the "real world" concerning the interval at which $P(t_1, \dots, t_n)$ is true, is partial. $P(t_1, \dots, t_n)$ thus describes multiple situations that are possibly the actual situation in the "real world". These possible situations described are precisely all situations in which $P(t_1, \dots, t_n)$ holds at an interval in $\mathcal{I}$. It is important to note that while a temporal literal states that it is true at exactly one interval in its interval set, this does not mean that it cannot be true at more than one interval in its interval set. This is only not what is expressed by the literal. If the same literal is true at multiple intervals, this can be expressed by multiple temporal literals which only differ by their interval sets.

The number of situations that are considered possible depends on the amount of information that is available. If for instance there is no information about at which interval something is true (only that it was true at some interval), then every situation in the "real world" in which this thing was true at some interval is a possible situation according to the corresponding temporal literal. The interval set of this temporal literal therefore should denote an interval set containing all intervals in the time span considered by the system.

If it is absolutely certain at which interval something is true, then only situations are considered possible in which this thing was true at this interval. The temporal term of the corresponding temporal literal therefore should denote an interval set containing only this interval. Multiple situations in which the exact same things are true at the same intervals are called a group of situations. A temporal literal containing only one interval in its interval set thus denotes a group of possible situations.

To illustrate the above, below are some examples of temporal literals and their meanings in natural language.

EXAMPLE 4.7 (TEMPORAL LITERALS AND THEIR MEANING)

The following are temporal literals with their corresponding interpretations in natural language:

- "Mary had a headache from 8 to 10" can be denoted by $headache(mary, \{(8,10)\})$. If it is in addition known that she had a headache from 12 to 15, then this can be denoted by an additional temporal literal $headache(mary, \{(12,15)\})$.
- If Mary had a fever and she is not sure whether it was from 8 to 10 or from 12 to 15, then this can be denoted by the temporal literal $fever(mary, \{(8,10), (12,15)\})$.
- "Mary did not have a stomach ache from 8 to 10 and she did not have a stomach ache from 12 to 15" can be denoted by $\neg stomachache(mary, \{(8,10)\})$ and $\neg stomachache(mary, \{(12,15)\})$.
- "Mary did not have a stomach ache from 5 to 6 or from 7 to 8" can be denoted by $\neg stomachache(mary, \{(5,6), (7,8)\})$.

In the next subsections, the proposed temporal argumentation logic is defined formally.

In (Allen, 1983), James Allen defines 13 relationships that can be used to describe any possible relationship between two intervals. Based on these relationships, some functions are defined that take an interval set and return an interval set of which the elements have a certain relationship with the intervals of the first. These functions can be used in temporal literals to denote which intervals are in an interval set and which intervals are not in this interval set. They do not add to the expressiveness of the language, but can be used as abbreviations.

DEFINITION 4.6 (FUNCTIONS ON INTERVAL SETS)

Consider an arbitrary interval set $\mathcal{I} = \{(i_1^-, i_1^+), \ldots, (i_k^-, i_k^+)\}$. The following unary functions on $\mathcal{I}$ return the set of all possible intervals that have a relationship with one of the intervals in $\mathcal{I}$ corresponding to the relationships in (Allen, 1983).

Function	Set
$\boldsymbol{equalTo}(\mathcal{I})$	$\mathcal{I}$
$\boldsymbol{before}(\mathcal{I})$	$\{(i_n^-, i_n^+) \mid i_n^+ < i_m^- \wedge i_m^- = \max(\{i^- \mid (i^-, i^+) \in \mathcal{I}\})\}$
$\boldsymbol{after}(\mathcal{I})$	$\{(i_n^-, i_n^+) \mid i_n^- > i_m^+ \wedge i_m^+ = \min(\{i^+ \mid (i^-, i^+) \in \mathcal{I}\})\}$
$\boldsymbol{during}(\mathcal{I})$	$\left\{(i_n^-, i_n^+) \middle\vert \begin{matrix} \begin{pmatrix} (i_n^- > i_m^- \wedge i_n^+ \leq i_m^+) \vee \\ (i_n^- \geq i_m^- \wedge i_n^+ < i_m^+) \end{pmatrix} \wedge \\ (i_m^-, i_m^+) \in \mathcal{I} \end{matrix} \right\}$
$\boldsymbol{contains}(\mathcal{I})$	$\left\{(i_n^-, i_n^+) \middle\vert \begin{matrix} \begin{pmatrix} (i_n^- < i_m^- \wedge i_n^+ \geq i_m^+) \vee \\ (i_n^- \leq i_m^- \wedge i_n^+ > i_m^+) \end{pmatrix} \wedge \\ (i_m^-, i_m^+) \in \mathcal{I} \end{matrix} \right\}$
$\boldsymbol{overlaps}(\mathcal{I})$	$\left\{(i_n^-, i_n^+) \middle\vert \begin{matrix} i_n^- < i_m^- \wedge i_n^+ > i_m^- \wedge \\ i_n^+ < i_m^+ \wedge (i_m^-, i_m^+) \in \mathcal{I} \end{matrix} \right\}$
$\boldsymbol{overlappedBy}(\mathcal{I})$	$\left\{(i_n^-, i_n^+) \middle\vert \begin{matrix} i_n^- > i_m^- \wedge i_n^- < i_m^+ \wedge \\ i_n^+ > i_m^+ \wedge (i_m^-, i_m^+) \in \mathcal{I} \end{matrix} \right\}$
$\boldsymbol{meets}(\mathcal{I})$	$\{(i_n^-, i_n^+) \mid i_n^+ \in start(\mathcal{I})\}$
$\boldsymbol{metBy}(\mathcal{I})$	$\{(i_n^-, i_n^+) \mid i_n^- \in end(\mathcal{I})\}$
$\boldsymbol{starts}(\mathcal{I})$	$\{(i_n^-, i_n^+) \mid i_n^+ < i_m^+ \wedge (i_n^-, i_m^+) \in \mathcal{I}\}$
$\boldsymbol{startedBy}(\mathcal{I})$	$\{(i_n^-, i_n^+) \mid i_n^+ > i_m^+ \wedge (i_n^-, i_m^+) \in \mathcal{I}\}$
$\boldsymbol{finishes}(\mathcal{I})$	$\{(i_n^-, i_n^+) \mid i_n^- > i_m^- \wedge (i_m^-, i_n^+) \in \mathcal{I}\}$
$\boldsymbol{finishedBy}(\mathcal{I})$	$\{(i_n^-, i_n^+) \mid i_n^- < i_m^- \wedge (i_m^-, i_n^+) \in \mathcal{I}\}$

REMARKS

The set returned by each function is finite, since the set $\mathcal{T}$ is defined to be finite and each end or starting point of an interval is an element of $\mathcal{T}$. This means that each variable in the definition above is implicitly bound by the restrictions on $\mathcal{T}$.

Below is an example of each of the functions on an interval set containing only one interval. The functions can of course be applied to interval sets containing more than one interval as well.

EXAMPLE 4.8 (FUNCTIONS ON INTERVAL SETS WITH ONE ELEMENT)

Suppose we know that Mary had a headache from 8 to 10, this knowledge can be represented by the predicate $headache(mary, \{(8{,}10)\})$.

The results of the functions defined above on the interval on which Mary had a headache are in the tables below.

Function	Set
$\boldsymbol{equalTo(\{(8,10)\})}$	$\{(8{,}10)\}$
$\boldsymbol{before(\{(8,10)\})}$	$\{(i^-, i^+) \mid i^+ < 8\}$
$\boldsymbol{after(\{(8,10)\})}$	$\{(i^-, i^+) \mid i^- > 10\}$
$\boldsymbol{during(\{(8,10)\})}$	$\left\{(i^-, i^+) \middle\| \begin{pmatrix} (i^- > 8 \wedge i^+ \leq 10) \vee \\ (i^- \geq 8 \wedge i^+ < 10) \end{pmatrix} \right\}$
$\boldsymbol{contains(\{(8,10)\})}$	$\left\{(i^-, i^+) \middle\| \begin{pmatrix} (i^- < 8 \wedge i^+ \geq 10) \vee \\ (i^- \leq 8 \wedge i^+ > 10) \end{pmatrix} \right\}$
$\boldsymbol{overlaps(\{(8,10)\})}$	$\left\{(i^-, i^+) \middle\| \begin{matrix} i^- < 8 \wedge i^+ > 8 \wedge \\ i^+ < 10 \end{matrix} \right\}$
$\boldsymbol{overlappedBy(\{(8,10)\})}$	$\left\{(i^-, i^+) \middle\| \begin{matrix} i^- > 8 \wedge i^- < 10 \wedge \\ i^+ > 10 \end{matrix} \right\}$
$\boldsymbol{meets(\{(8,10)\})}$	$\{(i^-, i^+) \mid i^+ = 8\}$
$\boldsymbol{metBy(\{(8,10)\})}$	$\{(i^-, i^+) \mid i^- = 10\}$
$\boldsymbol{starts(\{(8,10)\})}$	$\{(i^-, i^+) \mid i^+ < 10 \wedge i^- = 8\}$
$\boldsymbol{startedBy(\{(8,10)\})}$	$\{(i^-, i^+) \mid i^+ > 10 \wedge i^- = 8\}$
$\boldsymbol{finishes(\{(8,10)\})}$	$\{(i^-, i^+) \mid i^- > 8 \wedge i^+ = 10\}$
$\boldsymbol{finishedBy(\{(8,10)\})}$	$\{(i^-, i^+) \mid i^- < 8 \wedge i^+ = 10\}$

At times it may be useful to have certain information about intervals, such as their duration. Below some functions are defined with one interval as argument. In certain cases it can be useful as well to state that two intervals have a certain relation with each other. Functions to do this are additionally defined below. These functions again correspond with the relations described in (Allen, 1983).

DEFINITION 4.7 (FUNCTIONS ON INTERVALS)

Consider an arbitrary interval (i^-, i^+). The following unary functions on (i^-, i^+) are defined, such that they return a natural number denoting respectively the starting point, ending point or duration of (i^-, i^+):

Function	Returns
$\boldsymbol{start((i^-, i^+))}$	i^-
$\boldsymbol{end((i^-, i^+))}$	i^+
$\boldsymbol{duration((i^-, i^+))}$	$i^+ - i^-$

Consider arbitrary intervals (i_1^-, i_1^+) and (i_2^-, i_2^+). The following binary functions on (i_1^-, i_1^+) and (i_2^-, i_2^+), return the truth value $true$ if and only if (i_1^-, i_1^+) and (i_2^-, i_2^+) have a relationship corresponding to the relationships described in (Allen, 1983). Otherwise these functions return the value $false$.

Function	$true$ iff
$equals((i_1^-, i_1^+), (i_2^-, i_2^+))$	$i_1^- = i_2^- \wedge i_1^+ = i_2^+$
$before((i_1^-, i_1^+), (i_2^-, i_2^+))$	$i_1^+ < i_2^-$
$after((i_1^-, i_1^+), (i_2^-, i_2^+))$	$i_1^- > i_2^+$
$during((i_1^-, i_1^+), (i_2^-, i_2^+))$	$(i_1^- > i_2^- \wedge i_1^+ \leq i_2^+) \vee (i_1^- \geq i_2^- \wedge i_1^+ < i_2^+)$
$contains((i_1^-, i_1^+), (i_2^-, i_2^+))$	$(i_1^- < i_2^- \wedge i_1^+ \geq i_2^+) \vee (i_1^- \leq i_2^- \wedge i_1^+ > i_2^+)$
$overlaps((i_1^-, i_1^+), (i_2^-, i_2^+))$	$i_1^- < i_2^- \wedge i_1^+ > i_2^- \wedge i_1^+ < i_2^+$
$overlappedBy((i_1^-, i_1^+), (i_2^-, i_2^+))$	$i_1^- > i_2^- \wedge i_1^- < i_2^- \wedge i_1^+ > i_2^+$
$meets((i_1^-, i_1^+), (i_2^-, i_2^+))$	$i_1^+ = i_2^-$
$metBy((i_1^-, i_1^+), (i_2^-, i_2^+))$	$i_1^- = i_2^+$
$starts((i_1^-, i_1^+), (i_2^-, i_2^+))$	$i_1^- = i_2^- \wedge i_1^+ < i_2^+$
$startedBy((i_1^-, i_1^+), (i_2^-, i_2^+))$	$i_1^- = i_2^- \wedge i_1^+ > i_2^+$
$finishes((i_1^-, i_1^+), (i_2^-, i_2^+))$	$i_1^+ = i_2^+ \wedge i_1^- > i_2^-$
$finishedBy((i_1^-, i_1^+), (i_2^-, i_2^+))$	$i_1^+ = i_2^+ \wedge i_1^- < i_2^-$

REMARKS

The functions $after$, $contains$, $overlappedBy$, $metBy$, $startedBy$ and $finishedBy$ do not add to the expressiveness of the language, as they are the inverse functions of respectively $before$, $during$, $overlaps$, $meets$, $starts$ and $finishes$ and these functions can thus be used with their arguments switched. They are however added to increase usability.

Below is an example of each of the unary and binary functions from definition 4.7.

EXAMPLE 4.9 (FUNCTIONS ON INTERVALS)

Below is an example of each of the unary functions with the natural number they return.

Function	Returns
$start((9, 15))$	9
$end((9, 15))$	15
$duration((9, 15))$	$15 - 9 = 6$

Below are examples of each of the binary functions. In the examples in the left column $true$ is returned as a value, while in the examples in the right column the value $false$ is returned.

$\boldsymbol{true}$	$false$
$\boldsymbol{equals((1,2),(1,2))}$	$equals((1,2),(1,3))$
$\boldsymbol{before((5,7),(8,9))}$	$before((5,7),(6,9))$
$\boldsymbol{after((8,9),(5,7))}$	$after((6,9),(5,7))$
$\boldsymbol{during((4,9),(3,9))}$	$during((4,9),(5,9))$
$\boldsymbol{contains((3,9),(4,9))}$	$contains((5,9),(4,9))$
$\boldsymbol{overlaps((3,5),(4,6))}$	$overlaps((3,5),(4,5))$
$\boldsymbol{overlappedBy((4,6),(3,5))}$	$overlappedBy((4,5),(3,5))$
$\boldsymbol{meets((1,2),(2,5))}$	$meets((1,2),(3,5))$
$\boldsymbol{metBy((2,5),(1,2))}$	$metBy((3,5),(1,2))$
$\boldsymbol{starts((1,5),(1,6))}$	$starts((1,5),(1,4))$
$\boldsymbol{startedBy((1,6),(1,5))}$	$startedBy((1,4),(1,5))$
$\boldsymbol{finishes((4,5),(1,5))}$	$finishes((4,5),(1,6))$
$\boldsymbol{finishedBy((1,5),(4,5))}$	$finishedBy((1,6),(4,5))$

When temporal information is incomplete, but not completely missing, the information that is available can be used to define the interval set of a literal. To do this, the functions from the two definitions above may be used. Below are some examples of temporal literals containing these functions with their meanings in natural language.

EXAMPLE 4.10 (INCOMPLETE TEMPORAL INFORMATION)

Consider a case in which Mary had a stomachache which started and stopped before she had a fever. It is known when she had the fever, namely from 3 to 6. This knowledge can be represented by the literals $fever(mary,\{(3,6)\})$ and $stomachache(mary, before(\{(3,6)\}))$.

Suppose Mary had a headache with duration of 3 or 4 time units somewhere between 6 and 15. This can be represented by the literal $headache(mary,\mathcal{I})$, where

$$\mathcal{I} = \left\{ i \middle| \begin{array}{c} i \in during(\{(6,15)\}) \wedge (\\ duration(i) = 3 \vee \\ duration(i) = 4) \end{array} \right\}.$$

Suppose Mary had a fever from 3 to 6 and more than one time unit after the fever stopped, she got a fever again. During this second time she had a fever, she started to feel dizzy, which stopped after the fever stopped. The information above can be represented by the literals $fever(mary,\{(3,6)\})$, $fever(mary,\mathcal{H})$, where

$$\mathcal{H} = \left\{ (i^-, i^+) \middle| \begin{matrix} (i^-, i^+) \in after(\{(3,6)\}) \wedge \\ i^- > \mathrm{end}\big((3,6)\big) + 1 \end{matrix} \right\} \text{ and}$$
$dizzy(mary, \mathcal{K})$, where $\mathcal{K} = overlappedBy(\mathcal{J})$.

Complex interval sets, such as in the example above, can be used as facts in a system using the proposed logic. These facts can be from a database or can be given as input by the user. It is important to note that the building of these interval sets in facts is not part of a program using the proposed logic itself, but it is part of its input and therefore their realization is not of our concern here. In addition, facts are considered to be ground, this means that interval sets should not contain any variables when used as input.

In the next section, rules and derivations in the proposed temporal argumentation logic are defined and discussed.

4.3 RULES AND DERIVATIONS

Strict and defeasible rules are defined similar to (García & Simari, 2004). The only difference between the definitions of (García & Simari, 2004) and the definitions below is that the body of a rule should be a sequence in the proposed logic. The need for this deviation from the "original" definitions will become more apparent in the course of this section.

DEFINITION 4.8 (STRICT RULES)

A strict rule is an ordered pair, denoted "$Head \leftarrow Body$" whose first member $Head$, is a literal, and whose second member, $Body$ is a finite non-empty sequence of literals.

A strict rule with the head L_0 and body $\langle L_1, \dots, L_n \rangle$ can also be written as $L_0 \leftarrow L_1, \dots, L_n$ $(n > 0)$.

DEFINITION 4.9 (DEFEASIBLE RULES)

A defeasible rule is an ordered pair, denoted "$Head \prec Body$" whose first member $Head$, is a literal, and whose second member, $Body$ is a finite non-empty sequence of literals.

A defeasible rule with the head L_0 and body $\langle L_1, \dots, L_n \rangle$ can also be written as $L_0 \prec L_1, \dots, L_n$ $(n > 0)$.

The definition of a defeasible logic program is the same as the definition of (García & Simari, 2004). Their remark about which symbols denote variables is left out since this already has been discussed for the proposed logic.

DEFINITION 4.10 (DEFEASIBLE LOGIC PROGRAM) (GARCÍA & SIMARI, 2004)

A defeasible logic program $\mathcal{P}$, abbreviated de.l.p., is a possibly infinite set of facts, strict rules and defeasible rules. In a program $\mathcal{P}$, we will distinguish the subset Π of facts and strict rules and the subset Δ of defeasible rules. When required we will denote $\mathcal{P}$ as (Π, Δ).

Strict and defeasible rules are ground. However following the usual convention, some examples will use "schematic rules" with variables. Given a "schematic rule" R, $Ground(R)$ stands for the set of all ground instances of R. Given a de.l.p $\mathcal{P}$ with schematic rules, we define:

$$Ground(\mathcal{P}) = \bigcup_{R \in \mathcal{P}} Ground(R)$$

In medical diagnostic systems, it may sometimes be necessary to express temporal constraints in the body of a rule. In order to diagnose a disease, it may for instance be required that symptoms have occurred in a certain order or that they have lasted for a certain amount of time. Such properties and relations between the intervals at which symptoms should have been present can be expressed by using temporal constraints in a rule. Temporal constraints state that the intervals at which the literals in the body of a rule can be true should have certain properties or relations to each other. If they do, then the head of a rule that contains them may be derived, if they don't, then the head of such a rule may not be derived.

As discussed in the previous section, in the proposed logic it is possible to express partial temporal information. This is done by using temporal literals that denote which situations could possibly be the actual situation in the "real world". Since it is not always known which possible situation is the actual situation, it may not be certain that the literals in the body of a rule are true at intervals that have the right properties to meet the constraints. In some cases, of the possible situations expressed by a temporal literal, some have the right properties regarding the intervals, while some do not. It is then possible that the constraints in the body of a rule are met, but not certain. In such cases, the head of a rule may still be derived, but under the assumption that the actual situation is one of the possible situations expressed in the temporal literal that meets the constraints.

Temporal constraints are expressed as a set denoting all situations in which the intervals at which the literals in the body of a rule are true have those relationships and properties. The head of a rule containing a temporal constraint may only be derived if there is at least one possible situation according to the literals in its body in which the constraints are met. A possible situation meets the temporal constraint if it is denoted in the set representing the constraint.

To be able to define the way of expressing temporal constraints above, it is first necessary that possible situations according to single or multiple literals can be expressed more explicit and precise. The definitions below are used to do this.

DEFINITION 4.11 (TEMPORAL LITERAL SEQUENCES)

A finite sequence $\mathcal{L} = \langle L_1, \dots, L_n \rangle$, where $L_1, \dots, L_n$ are temporal literals is called a temporal literal sequence.

Below are some examples of temporal literal sequences.

EXAMPLE 4.11 (TEMPORAL LITERAL SEQUENCES)

The following sequences are temporal literal sequences:

- $\langle P(c, \mathcal{I}) \rangle$
- $\langle P(c, \mathcal{I}), P(c, \mathcal{H}) \rangle$
- $\langle P(c_1, \mathcal{I}), Q(c_2, \mathcal{H}) \rangle$
- $\left\langle \begin{matrix} fever(mary, \{(5,8), (15,19)\}), \\ headache(mary, after(\{15,19\})) \end{matrix} \right\rangle$

The following are not temporal literal sequences:

- $\langle P(c) \rangle$
- $\langle P(c, \mathcal{I}), Q(x, \mathcal{H}) \rangle$
- $\langle P(c, \mathcal{I}), \mathcal{I} \rangle$

The information about which situations are possibly the actual situation in the “real world” on basis of multiple temporal literals is captured in a temporal literal sequence. According to the information in a temporal literal sequence, all situations that are not contradictory in which each of the literals in it is true at an interval in its temporal term are possible. A temporal literal sequence by itself is not sufficient to describe specific possible situations on basis of the information in it. To denote specific possible situations, temporal literal sequences are combined with configurations.

DEFINITION 4.12 (CONFIGURATIONS)

A finite sequence $\mathcal{C} = \langle (i_1^-, i_1^+), \dots, (i_n^-, i_n^+) \rangle$, where $(i_1^-, i_1^+), \dots, (i_n^-, i_n^+)$ are intervals is called a configuration.

Below are some examples of configurations.

EXAMPLE 4.12 (CONFIGURATIONS)

The following sequences are configurations:

- $\langle (i^-, i^+) \rangle$
- $\langle (i^-, i^+), (i^-, i^+) \rangle$
- $\langle (i_1^-, i_1^+), (i_2^-, i_2^+) \rangle$

Temporal literal sequences and configurations can be combined to denote specific possible situations. The configurations are used to denote the intervals at which the temporal literals in a temporal literal sequence are true in these situations expressed. To denote possible situations, the intervals in the configuration should be elements of the interval sets of the corresponding literals and they should be in the right order. In addition, the situations denoted should not be contradictory; otherwise the actual situation in the "real world" cannot possibly be one of them and the situation denoted is not possible.

The notion of matching temporal literals is used to prevent that a combination of a temporal literal sequence and a configuration describes situations that are contradictory. Situations are contradictory if the same thing is true and false at overlying intervals. Overlying intervals are defined in definition 4.14.

DEFINITION 4.13 (MATCHING TEMPORAL LITERALS)

Let $P(t_1, \dots, t_n)$ and $P(u_1, \dots, u_n)$ be temporal literals with temporal terms t_i and u_i. $P(t_1, \dots, t_n)$ and $P(u_1, \dots, u_n)$ are said to match if and only for every natural number j between 0 and n that is not equal to i, $t_j = u_j$.

Below are some examples of matching temporal literals.

EXAMPLE 4.13 (MATCHING TEMPORAL LITERALS)

The following temporal literals match:

- $P(c, \mathcal{I})$ and $P(c, \mathcal{I})$
- $P(c, \mathcal{I})$ and $P(c, \mathcal{H})$
- $fever(mary, \{(1,3), (4,7)\})$ and $fever(mary, \{(10,14)\})$

The following temporal literals do not match:

- $P(c_1, \mathcal{I})$ and $P(c_2, \mathcal{H})$, where $c_1 \neq c_2$
- $Q(c, \mathcal{I})$ and $P(c, \mathcal{H})$
- $fever(mary, \{(1,3), (4,7)\})$ and $fever(jack, \{(10,14)\})$

Two temporal literals match if they are the same except for their temporal term. Two matching temporal literals thus denote situations in which the same relation between the same arguments holds, but possibly at different intervals. If a literal matches the negation of another literal, this means that one of them describes situations in which a relationship between arguments holds at one or more intervals, while the other describes situations in which the same relationships between the same arguments does not hold at certain intervals. A group of situations is contradictory and thus does not contain any possible situations if in it the same relationship between the same arguments holds and does not hold at a overlying interval. Such a situation can be denoted by two temporal literals of which one matches the negation of the other if they both contain the same interval in their interval set.

Intervals are overlying if they have at least one point in time in common. This is however left implicit and the functions from definition 4.7 are used to define the notion of a overlying interval.

DEFINITION 4.14 (OVERLYING INTERVALS)

An interval (i_i^-, i_i^+) overlies with an interval (i_j^-, i_j^+) if and only if one of the following functions returns the value $true$:

- $equalTo\left((i_i^-, i_i^+), (i_j^-, i_j^+)\right)$, or
- $during\left((i_i^-, i_i^+), (i_j^-, i_j^+)\right)$, or
- $contains\left((i_i^-, i_i^+), (i_j^-, i_j^+)\right)$, or
- $overlaps\left((i_i^-, i_i^+), (i_j^-, i_j^+)\right)$, or
- $overlappedBy\left((i_i^-, i_i^+), (i_j^-, i_j^+)\right)$, or
- $starts\left((i_i^-, i_i^+), (i_j^-, i_j^+)\right)$, or
- $startedBy\left((i_i^-, i_i^+), (i_j^-, i_j^+)\right)$, or
- $finishes\left((i_i^-, i_i^+), (i_j^-, i_j^+)\right)$, or
- $finishedBy\left((i_i^-, i_i^+), (i_j^-, i_j^+)\right)$

Below are some examples of overlying intervals.

EXAMPLE 4.14 (OVERLYING INTERVALS)

The following intervals are overlying:

- $(1,3)$ and $(1,3)$
- $(1,3)$ and $(2,7)$
- $(1,3)$ and $(3,4)$
- $(9,17)$ and $(6,10)$
- $(9,17)$ and $(10,15)$

The following intervals are not overlying:

- $(1,3)$ and $(4,7)$
- $(1,3)$ and $(10,16)$
- $(9,17)$ and $(6,8)$

Temporal literal sequences and configurations are combined in a configuration pair to denote exactly one group of situations in which the literals in the temporal literal sequence are true at the intervals in the configuration. Contradictory situations are not considered possible and therefore cannot be described by a configuration pair. To exclude the pairs of temporal literal sequences and configurations describing groups of impossible situations, the third requirement is added to the definition.

DEFINITION 4.15 (CONFIGURATION PAIRS)

Let $\mathcal{L}$ be a temporal literal sequence and let $\mathcal{C}$ be a configuration. The ordered pair $\langle \mathcal{L}, \mathcal{C} \rangle$ is a configuration pair if and only if:

1. $|\mathcal{L}| = |\mathcal{C}|$, and
2. if $\imath_j \in \mathcal{C}$, then $\imath_j \in \mathcal{I}_j$, where $\mathcal{I}_j$ is the temporal term of $L_j \in \mathcal{L}$[9], and
3. if there are $L_j, L_k \in \mathcal{L}$ such that L_j matches the negation of L_k or vice versa, then $\imath_j, \imath_k \in \mathcal{C}$ do not overlie

REMARKS

A configuration pair is used to denote exactly one group of situations in the "real world" that contains possible situations according to the information in its temporal literal sequence. Let $\langle\langle L_1, \dots, L_n\rangle, \langle \imath_1, \dots, \imath_n\rangle\rangle$ be a configuration pair. This configuration pair denotes a group of situations in which L_1 is true at interval $\imath_1$, L_2 is true at interval $\imath_2$, …, and L_n is true at interval $\imath_n$. Since each $\imath_i$ should be an element of the interval set of L_i and contradictory situations are excluded by the third requirement, a configuration pair clearly denotes groups of situations that are possible according to the literals in its temporal literal sequence.

In a configuration pair, each of the intervals in a configuration depends in a way on the literal that has the same position in the temporal literal sequence as the interval has in its configuration. The interval should, to be precise, be an element of the interval set denoted by the temporal term of this temporal literal. This means that the first interval in a configuration should be an element of the interval set denoted by the temporal term of the first temporal literal in the temporal literal sequence, the second interval should be an element of the interval set denoted by the temporal term of the second interval set, and so on. In the following configuration pair intervals have the same color as the temporal literals on which they depend: $\langle\langle L_1, L_2, L_3 \dots, L_n\rangle, \langle \imath_1, \imath_2, \imath_3 \dots, \imath_n\rangle\rangle$.

Some examples of configuration pairs are below.

[9] For elements of sequences, the subscript denotes the position in the set, e.g. $L_j \in \mathcal{L}$ denotes the jth element of $\mathcal{L}$ and $(i_j^-, i_j^+) \in \mathcal{C}$ denotes the jth element of $\mathcal{C}$.

EXAMPLE 4.15 (CONFIGURATION PAIRS)

The following are configuration pairs:

- $\langle\langle P(c,\{i\}\rangle,\langle i\rangle\rangle$
- $\langle\langle P(c_1,\{i_1\}),P(c_2,\{i_2\})\rangle,\langle i_1,i_2\rangle\rangle$
- $\langle\langle P(c_1,\{i_1,i_2\}),Q(c_2,\{i_3\})\rangle,\langle i_1,i_3\rangle\rangle$
- $\langle\langle P(c_1,\{i_1,i_2\}),Q(c_2,\{i_3\})\rangle,\langle i_2,i_3\rangle\rangle$
- $\langle\langle P(c_1,\{(1,6),(3,5)\}),Q(c_2,\{(9,20),(10,30)\})\rangle,\langle(3,5),(9,20)\rangle\rangle$
- $\langle\langle P\big(c,before(\{(15,20)\})\big)\rangle,\langle(2,4)\rangle\rangle$
- $\langle\langle P(c,\mathcal{I}),\neg P(c,\mathcal{H})\rangle,\langle i_1,i_2\rangle\rangle$, where $i_1\in\mathcal{I}$ and $i_2\in\mathcal{H}$ do not overlie

The following are ordered pairs that are not configuration pairs:

- $\langle\langle P(c,\{i_1\}\rangle,\langle i_2\rangle\rangle$, where $i_1\neq i_2$
- $\langle\langle P(c_1,\{i_1\}),P(c_2,\{i_2\})\rangle,\langle i_2,i_1\rangle\rangle$, where $i_1\neq i_2$
- $\langle\langle P(c_1,\{i_1,i_2\}),Q(c_2,\{i_3\})\rangle,\langle i_1,i_2,i_3\rangle\rangle$
- $\langle\langle P\big(c,before(\{(15,20)\})\big)\rangle,\langle(25,30)\rangle\rangle$
- $\langle\langle P(c,\mathcal{I}),\neg P(c,\mathcal{H})\rangle,\langle i,i\rangle\rangle$

In a configuration pair, the configuration may contain any of the intervals of the interval sets of the corresponding literals in the temporal literals set, as long as no group of contradictory situations is expressed. Therefore, every group of possible situations according to a set of temporal literals can be denoted by a configuration pair.

As an example, let $headache(mary,\{(1,3),(4,6)\})$ denote that "Mary had a headache at interval (1,3) or at interval (4,6)" and let $fever(mary,\{(2,6),(9,11)\})$ denote that "Mary had a fever at interval (2,6) or at interval (9,11)". From these literals, the temporal literal sequence $\left\langle\begin{matrix}headache(mary,\{(1,3),(4,6)\}),\\ fever(mary,\{(2,6),(9,11)\})\end{matrix}\right\rangle$ can be composed. According to the definition above, this temporal literal sequence can form a configuration pair with configuration $\langle(1,3),(2,6)\rangle$, viz. $\left\langle\left\langle\begin{matrix}headache(mary,\{(1,3),(4,6)\}),\\ fever(mary,\{(2,6),(9,11)\})\end{matrix}\right\rangle,\langle(1,3),(2,6)\rangle\right\rangle$. This configuration pair denotes the group of situations in which Mary had a headache at interval $(1,3)$ and she had a fever at interval $(2,6)$.

The group of situations denoted by the configuration pair above is clearly a group of possible situations according to the information in the literals in its temporal literal sequence. Of course, other groups of situations are considered possible based on the same information. They each can be denoted by a configuration pair with the same temporal literal sequence, but a different configuration. Examples of other configurations that can form a pair with the temporal literal sequence above are $\langle(4,6),(2,6)\rangle$ and $\langle(1,3),(9,11)\rangle$.

There are also configurations with which the temporal literal sequence above cannot form a pair. An example is configuration $\langle(4,6),(2,9)\rangle$, since $(2,9)$ is not an element of the temporal term of $fever(mary,\{(2,6),(9,11)\})$. Note also that if it were possible that $\langle(4,6),(2,9)\rangle$

formed a configuration pair with the temporal literal sequence above, this configuration pair would not denote situations that are possible according to the information in its temporal literal sequence.

To define temporal constraints and later on assumptions, it is in addition necessary to be able to denote multiple groups of possible situations on basis of the information in a temporal literal sequence. This can be done by using multi-configuration pairs. Multi-configuration pairs are very similar to configuration pairs, the difference between them is that instead of a configuration, a multi-configuration pair contains a set of configurations.

DEFINITION 4.16 (CONFIGURATION SETS)

A set $\mathcal{D} = \{\mathcal{C}_1, \dots, \mathcal{C}_n\}$, where $\mathcal{C}_1, \dots, \mathcal{C}_n$ are configurations with the same cardinality, is called a configuration set.

Some examples of configuration sets are below.

EXAMPLE 4.16 (CONFIGURATION SETS)

The following sets are configuration sets:

- $\{\langle i \rangle\}$
- $\left\{\begin{matrix}\langle i_1, i_1 \rangle, \langle i_2, i_2 \rangle, \\ \langle i_3, i_3 \rangle, \langle i_4, i_4 \rangle\end{matrix}\right\}$
- $\left\{\begin{matrix}\langle i_1, i_2, i_3, i_4 \rangle, \\ \langle i_5, i_6, i_7, i_8 \rangle\end{matrix}\right\}$
- $\left\{\begin{matrix}\langle (1,2), (5,6) \rangle, \langle (3,4), (5,6) \rangle, \\ \langle (1,2), (7,8) \rangle, \langle (3,4), (7,8) \rangle\end{matrix}\right\}$
- $\mathcal{I} \times \mathcal{J}$

The following are not configuration sets:

- $\{\langle i_1, i_2 \rangle, \langle i_1 \rangle\}$
- $\left\{\begin{matrix}\langle (1,2), (5,6) \rangle, \langle (3,4), (5,6) \rangle, \\ \langle (1,2), (7,8) \rangle, \langle (1,2), (3,4), (7,8) \rangle\end{matrix}\right\}$
- $\mathcal{I}$

The definition of a multi-configuration pair is below.

DEFINITION 4.17 (MULTI-CONFIGURATION PAIRS)

Let $\mathcal{L}$ be a temporal literal sequence and let $\mathcal{D}$ be a configuration set. The ordered pair $\langle \mathcal{L}, \mathcal{D} \rangle$ is said to be a multi-configuration pair, if and only if for every configuration $\mathcal{C} \in \mathcal{D}$, $\langle \mathcal{L}, \mathcal{C} \rangle$ is a configuration pair.

Below are some examples of multi-configuration pairs.

EXAMPLE 4.17 (MULTI-CONFIGURATION PAIRS)

The following are multi-configuration pairs:

- $\langle\langle P(c,\{\imath\})\rangle,\{\langle\imath\rangle\}\rangle$
- $\langle\langle P(c,\{\imath_1,\imath_2\}),Q(c,\{\imath_3,\imath_4\})\rangle\{\langle\imath_1,\imath_3\rangle,\langle\imath_2,\imath_3\rangle\}\rangle$
- $\left\langle\begin{matrix}\langle P(c,\{(1,2),(3,4)\}),Q(c,\{(5,6)\}),R(c,\{(7,8)\})\rangle,\\ \{\langle(1,2),(5,6),(7,8)\rangle,\langle(3,4),(5,6),(7,8)\rangle\}\end{matrix}\right\rangle$

The following pairs are not multi-configuration pairs:

- $\langle\langle P(c,\{\imath_1\}\rangle\{\langle\imath_2\rangle\}\rangle$, where $\imath_1 \neq \imath_2$
- $\langle\langle P(c,\{\imath_1,\imath_2\}),Q(c,\{\imath_3,\imath_4\})\rangle,\{\langle\imath_1,\imath_3\rangle,\langle\imath_2\rangle\}\rangle$
- $\left\langle\begin{matrix}\langle P(c,\{(1,2),(3,4)\}),Q(c,\{(5,6)\}),R(c,\{(7,8)\})\rangle,\\ \{\langle(1,2),(5,6),(7,8)\rangle\langle(3,4),(5,6),(1,2)\rangle\}\end{matrix}\right\rangle$

A multi-configuration pair denotes multiple groups of possible situations based on the information in its temporal literal sequence. Each group of possible situations denoted by a multi-configuration pair can be denoted by a configuration pair containing the same temporal literal sequence and a configuration from its configuration set. The number of the groups of situations denoted by a multi-configuration pair is the same as the number of unique elements in its configuration set.

Consider the temporal literal sequence at page 58 once again. This temporal literal sequence can form a multi-configuration pair with configuration set $\{\langle(4,6),(2,6)\rangle,\langle(1,3),(9,11)\rangle\}$. This multi-configuration pair denotes two groups of situations, namely a group of situations in which "Mary has a headache at $(4,6)$ and Mary has a fever at $(2,6)$" and a group of situations in which "Mary has a headache at $(1,3)$ and Mary has a fever at $(9,11)$". This multi-configuration pair does for instance not denote a group of situations in which "Mary has a headache at $(4,6)$ and Mary has a fever at $(9,11)$", since the configuration $\langle(4,6),(9,11)\rangle$ is not part of the configuration set of this multi-configuration pair.

A complete multi-configuration pair denotes all possible situations according the temporal literals in its temporal literal sequence. Each of the groups of situations denoted by a multi-configuration pair is denoted by a configuration in its configuration set. The configuration set of a complete multi-configuration pair should therefore contain all configurations with which its temporal literal sequence can form a configuration pair.

DEFINITION 4.18 (COMPLETE MULTI-CONFIGURATION PAIRS)

Let $\langle\mathcal{L},\mathcal{D}\rangle$ be a multi-configuration pair. $\langle\mathcal{L},\mathcal{D}\rangle$ is a complete multi-configuration pair if and only if for every configuration $\mathcal{C}$, if $\langle\mathcal{L},\mathcal{C}\rangle$ is a configuration pair, then $\mathcal{C}\in\mathcal{D}$.

REMARKS

Let $\mathcal{L} = \langle L_1, \dots, L_n \rangle$ be a temporal literal sequence and $\mathcal{J}_1, \dots, \mathcal{J}_n$ be the interval sets denoted by the temporal terms of $L_1, \dots, L_n$ respectively. Let $\langle \mathcal{L}, \mathcal{D} \rangle$ be a complete multi-configuration set. If there is no configuration $\mathcal{C}$, such that $\langle \mathcal{L}, \mathcal{C} \rangle$ does meet the first two requirements of definition 4.15, but not the third, then $\mathcal{D}$ is the n-ary Cartesian product of $\mathcal{J}_1, \dots, \mathcal{J}_n$, i.e. $\mathcal{D} = \mathcal{J}_1 \times \dots \times \mathcal{J}_n$.

As an example, consider the following temporal literal sequence:
$\mathcal{L}_1 = \langle fever(mary, \{(1,2), (3,4)\}), \neg fever(mary, \{(5,6)\}) \rangle$, the configuration set of its complete multi-configuration pair is $\{\langle (1,2), (5,6) \rangle, \langle (3,4), (5,6) \rangle\}$, which is clearly equal to $\{(1,2), (3,4)\} \times \{(5,6)\}$. Now consider temporal literal sequence
$\mathcal{L}_2 = \langle fever(mary, \{(1,2), (3,4)\}), \neg fever(mary, \{(3,4)\}) \rangle$. The configuration set of the complete multi-configuration pair of $\mathcal{L}_2$ is $\{\langle (1,2), (3,4) \rangle\}$. Configuration $\langle (3,4), (3,4) \rangle$ does not form a configuration pair with $\mathcal{L}_2$ since it describes a group of situations that is contradictory and thus does not meet the third requirement of definition 4.15. It is thus excluded from the configuration set of the complete multi-configuration pair of $\mathcal{L}_2$. It can be observed easily that for this reason, the complete multi-configuration pair of $\mathcal{L}_2$ is not the Cartesian product of the interval sets denoted by the temporal terms of the temporal literals contained in it.

Below are some examples of complete and incomplete multi-configuration pairs.

EXAMPLE 4.18 (COMPLETE MULTI-CONFIGURATION PAIRS)

The following multi-configuration pairs are complete:

- $\langle \langle P(c, \{\imath\}) \rangle, \{\langle \imath \rangle\} \rangle$
- $\langle \langle P(c, \{\imath\}), \neg P(c, \{\imath\}) \rangle, \emptyset \rangle$
- $\langle \langle P(c, \{\imath_1, \imath_2\}), Q(c, \{\imath_3, \imath_4\}) \rangle, \left\{ \begin{matrix} \langle \imath_1, \imath_3 \rangle, \langle \imath_2, \imath_3 \rangle, \\ \langle \imath_1, \imath_4 \rangle, \langle \imath_2, \imath_4 \rangle \end{matrix} \right\} \rangle$
- $\left\langle \begin{matrix} \langle P(c, \{(1,2), (3,4)\}), Q(c, \{(5,6)\}), R(c, \{(7,8)\}) \rangle, \\ \{\langle (1,2), (5,6), (7,8) \rangle \langle (3,4), (5,6), (1,2) \rangle\} \end{matrix} \right\rangle$
- $\langle \langle P(c, \mathcal{J}), Q(c, \mathcal{H}), R(c, \mathcal{K}) \rangle, \mathcal{J} \times \mathcal{H} \times \mathcal{K} \rangle$

The following are multi-configuration pairs that are not complete:

- $\langle \langle P(c, \{\imath_1\} \rangle, \{\langle \imath_2 \rangle, \langle \imath_3 \rangle\} \rangle$, where $\imath_2 \neq \imath_3$
- $\langle \langle P(c, \{\imath_1, \imath_2\}), Q(c, \{\imath_3, \imath_4\}) \rangle, \left\{ \begin{matrix} \langle \imath_1, \imath_3 \rangle, \langle \imath_2, \imath_3 \rangle, \\ \langle \imath_1, \imath_4 \rangle, \langle \imath_2, \imath_5 \rangle \end{matrix} \right\} \rangle$, where $\imath_5 \neq \imath_4$

- $\langle \substack{\langle P(c,\{(1,2),(3,4)\}), Q(c,\{(5,6)\}), R(c,\{(7,8)\})\rangle, \\ \{\langle(1,2),(5,6),(7,8)\rangle\}} \rangle$
- $\langle\langle P(c,\mathcal{I}), Q(c,\mathcal{H}), R(c,\mathcal{K})\rangle, \mathcal{I}\times\mathcal{K}\times\mathcal{H}\rangle$, where $\mathcal{H}\neq\mathcal{K}$

To explain the notion of a complete multi-configuration pair, let $headache(mary,\{(1,3),(4,6)\})$ once again denote that "Mary had a headache at interval $(1,3)$ or at interval $(4,6)$" and let $fever(mary,\{(2,6),(9,11)\})$ denote that "Mary had a fever at interval $(2,6)$ or at interval $(9,11)$". From these literals, the temporal literal sequence $\mathcal{L} = \langle headache(mary,\{(1,3),(4,6)\}), fever(mary,\{(2,6),(9,11)\})\rangle$ can be composed.

According to the information in the temporal literal sequence above there are four groups of situations possible in the "real world", viz.:

- Mary had a had a headache at interval $(1,3)$ and Mary had a fever at interval $(2,6)$
- Mary had a had a headache at interval $(1,3)$ and Mary had a fever at interval $(9,11)$
- Mary had a had a headache at interval $(4,6)$ and Mary had a fever at interval $(2,6)$
- Mary had a had a headache at interval $(4,6)$ and Mary had a fever at interval $(9,11)$

The possible situations in the real world described above can each be denoted by a configuration pair, namely $\langle\mathcal{L},\langle(1,3),(2,6)\rangle\rangle$, $\langle\mathcal{L},\langle(1,3),(9,11)\rangle\rangle$, $\langle\mathcal{L},\langle(4,6),(2,6)\rangle\rangle$ and $\langle\mathcal{L},\langle(4,6),(9,11)\rangle\rangle$ respectively. Since these configuration pairs are all possible configuration pairs on basis of $\mathcal{L}$, the multi-configuration pair $\langle\mathcal{L},\{\langle(1,3),(2,6)\rangle,\langle(1,3),(9,11)\rangle,\langle(4,6),(2,6)\rangle,\langle(4,6),(9,11)\rangle\}\rangle$ is complete. Note that the configuration set of this multi-configuration pair contains every configuration of the configuration pairs above.

An example of a multi-configuration pair that is not complete is $\langle\mathcal{L},\{\langle(1,3),(2,6)\rangle,\langle(1,3),(9,11)\rangle,\langle(4,6),(2,6)\rangle\}\rangle$. It is not complete because configuration pair $\langle\mathcal{L},\langle(4,6),(9,11)\rangle\rangle$ exists and $\langle(4,6),(9,11)\rangle$ is not an element of its configuration set. Clearly this multi-configuration pair does not denote all possible situations on basis of the information in $\mathcal{L}$.

Using the definitions above, it is possible to express one or more specific groups of possible situations on basis of single or multiple temporal literals explicitly. By using the definition below, it is possible to denote the groups of situations that are possible according to the temporal literals in the body of a rule.

DEFINITION 4.19 (TEMPORAL LITERAL SEQUENCES OF RULES)

Let R be a strict or defeasible rule with body $Body$. Temporal literal sequence $\mathcal{L}$ is said to be the temporal literal sequence of R if and only if:

- For each $L_i \in \mathcal{L}$, there is a $L_j \in Body$ and $L_i = L_j$
- For each $L_i \in Body$ that is a temporal literal, there is a $L_j \in \mathcal{L}$ and $L_i = L_j$
- For each $L_i, L_j \in Body$ that are temporal literals, if $i < j$, then there are $L_k, L_l \in \mathcal{L}$ and $L_i = L_k$ and $L_j = L_l$ and $k < l$.

Below are some examples of rules and their temporal literal sequences.

EXAMPLE 4.19 (TEMPORAL LITERAL SEQUENCES OF RULES)

The following are rules with their temporal literal sequences:

- $P(c_1) \leftarrow Q(c_1, c_2), R(c_1, \mathcal{I})$ and $\langle R(c_1, \mathcal{I}) \rangle$
- $P(c) \prec Q(c, \mathcal{I}), R(c, \mathcal{H}), \mathcal{I} \times \mathcal{H} \cap \mathcal{D} \neq \emptyset$ and $\langle Q(c, \mathcal{I}), R(c, \mathcal{H}) \rangle$
- $P(c) \prec Q(c, \mathcal{I}), R(c), S(c), T(\mathcal{H})$ and $\langle Q(c, \mathcal{I}), T(\mathcal{H}) \rangle$

The following are rules with temporal literal sequences that are not their temporal literal sequences:

- $P(c) \prec Q(c, \mathcal{I}), R(c, \mathcal{H})$ and $\langle R(c, \mathcal{H}), Q(c, \mathcal{I}) \rangle$
- $P(c) \leftarrow Q(c, \mathcal{I}), R(c, \mathcal{H}), \mathcal{I} \times \mathcal{H} \cap \mathcal{D} \neq \emptyset$ and $\langle Q(c, \mathcal{I}) \rangle$
- $P(c) \leftarrow Q(c, \mathcal{I}), R(c), S(c), T(\mathcal{H})$ and $\langle Q(c, \mathcal{I}), S(\mathcal{H}) \rangle$

By definition 4.19, the temporal literal sequence of a rule is a temporal literal sequence containing every temporal literal in the body of the rule in the same order. The complete multi-configuration pair of the temporal literal sequence of a rule denotes all possible situations on basis of the information in the body of the rule. A temporal constraint is expressed as a configuration set that is such that every configuration that contains intervals that have the right properties and relations is in it. It can be verified whether there is a possible situation on basis of the temporal literal sequence of a rule that meets the constraints, by checking whether the intersection of the configuration set expressing the constraint and the configuration set of the complete multi-configuration pair of the temporal literal sequence of the rule is empty. This can be done by including temporal equations in the body of a rule.

DEFINITION 4.20 (TEMPORAL EQUATIONS)

A temporal equation is a literal of the form $\mathcal{D}_1 \cap \mathcal{D}_2 \neq \emptyset$, where $\mathcal{D}_1$ and $\mathcal{D}_2$ are configuration sets.

$\mathcal{D}_1$ and $\mathcal{D}_2$ are said to be the configuration sets of $\mathcal{D}_1 \cap \mathcal{D}_2 \neq \emptyset$.

EXAMPLE 4.20 (TEMPORAL EQUATIONS)

The following are temporal equations:

- $\{\langle i_1, i_2\rangle\langle i_1, i_3\rangle\} \cap \{\langle i_1, i_2\rangle\langle i_1, i_4\rangle\} \neq \emptyset$
- $before(\mathcal{I}) \times \mathcal{H} \cap \mathcal{D} \neq \emptyset$
- $\{\langle(1,2),(3,4)\rangle, \langle(5,6),(7,8)\rangle\} \cap \{\langle(1,2),(3,4)\rangle\} \neq \emptyset$
- $\mathcal{I} \times \mathcal{H} \times \mathcal{K} \cap \left\{\langle i_1, i_2, i_3\rangle \middle| \begin{matrix} before(i_1, i_2) \wedge \\ during(i_3, i_2) \end{matrix}\right\} \neq \emptyset$

The following are not temporal equations:

- $\mathcal{D} \neq \emptyset$
- $\mathcal{D}_1 \cap \mathcal{D}_2 = \emptyset$

Just including a temporal equation in the body of a rule is not enough to conduct sound reasoning. This is partly due to the fact that if it is possible but not certain on basis of the information in the body of a rule that constraints are met, it needs to be assumed that the actual situation in the “real world” is one of the possible situations that do meet the constraints. Such an assumption can be expressed by a multi-configuration pair expressing all possible situations that do meet the constraints on basis of the information in the body of a rule. Determining what configurations should be in the configuration set of such a multi-configuration pair would for instance be very hard if a rule contains multiple temporal equations or if the temporal equation is such that not all possible situations are checked. For these reasons, the notion of a temporal rule is defined. A temporal rule has exactly the right properties to be able to determine the exact assumption that needs to be made when deriving its head.

DEFINITION 4.21 (TEMPORAL RULES)

Let R be a strict or defeasible rule with body $Body$ and let $\mathcal{L}$ be its temporal literal sequence. R is a temporal rule if and only if:

- There is exactly one literal $L \in Body$, such that L is a temporal equation, and
- If $(\mathcal{D}_1 \cap \mathcal{D}_2 \neq \emptyset) \in Body$ is a temporal equation, then $\langle\mathcal{L}, \mathcal{D}_1\rangle$ or $\langle\mathcal{L}, \mathcal{D}_2\rangle$ is a complete multi-configuration pair.

If L is a temporal equation in the body of R, then it is said that L is the temporal equation of R.

Below are some examples of temporal and non-temporal rules.

EXAMPLE 4.21 (TEMPORAL RULES)

The following rules are temporal rules:

- $P(c) \leftarrow Q(c,\mathcal{I}), R(c,\mathcal{H}), \mathcal{I} \times \mathcal{H} \cap \mathcal{D} \neq \emptyset$
- $P(c) \prec Q(c,\mathcal{I}), S(c), R(c,\mathcal{H}), \mathcal{I} \times \mathcal{H} \cap \mathcal{D} \neq \emptyset$

- $P(c) \leftarrow Q(c,\{(7{,}10),(19{,}20)\}), R(c,\{(12{,}15)\}),$ $\{\langle(7{,}10),(12{,}15)\rangle \times \langle(19{,}20),(12{,}15)\rangle\}$ $\cap \{\langle i_1, i_2\rangle | before(i_1,i_2)\} \neq \emptyset$

The following are not temporal rules:

- $P(c) \leftarrow Q(c,\mathcal{I}), R(c,\mathcal{H})$
- $P(c) \prec Q(c,\mathcal{I}), S(c), R(c,\mathcal{H}), \mathcal{I} \times \mathcal{H} \cap \mathcal{D}_1 \neq \emptyset, \mathcal{I} \times \mathcal{H} \cap \mathcal{D}_2 \neq \emptyset$
- $P(c) \leftarrow Q(c,\{(7{,}10),(19{,}20)\}), R(c,\{(12{,}15)\}), \{\langle(7{,}10),(12{,}15)\rangle\} \cap \{\langle i_1, i_2\rangle | before(i_1,i_2)\} \neq \emptyset$

It is important to know which argument of the intersection in a temporal equation denotes the temporal constraints. Temporal constraints are therefore defined below.

DEFINITION 4.22 (TEMPORAL CONSTRAINTS)

Let R be a temporal rule and let $\mathcal{L}$ be its temporal literal sequence. A configuration set $\mathcal{B}$ is said to be the temporal constraint of R if and only if $\mathcal{D} \cap \mathcal{B} \neq \emptyset$ is the temporal equation of R and $\langle \mathcal{L}, \mathcal{D} \rangle$ is a complete multi-configuration pair.

Below are some examples of temporal rules with their temporal constraints.

EXAMPLE 4.22 (TEMPORAL CONSTRAINTS)

The following are temporal rules and their temporal constraints.

- $P(c) \leftarrow Q(c,\mathcal{I}), R(c,\mathcal{H}), \mathcal{D} \cap \mathcal{I} \times \mathcal{H} \neq \emptyset$ and $\mathcal{D}$
- $P(c) \prec Q(c,\mathcal{I}), S(c), R(c,\mathcal{H}), \mathcal{I} \times \mathcal{H} \cap \mathcal{D} \neq \emptyset$ and $\mathcal{D}$
- $P(c) \leftarrow Q(c,\{(7{,}10),(19{,}20)\}), R(c,\{(12{,}15)\}),$ $\{\langle(7{,}10),(12{,}15)\rangle \times \langle(19{,}20),(12{,}15)\rangle\} \cap$ $\{\langle i_1, i_2\rangle | before(i_1,i_2)\} \neq \emptyset$ and $\{\langle i_1, i_2\rangle | before(i_1,i_2)\}$

As an illustration of the way temporal constraints are expressed in the proposed logic, below are some examples of temporal rules and their meanings in natural language:

- $cat\_allergy(mary) \leftarrow runny\_nose(mary,\mathcal{I}), contact\_cats(mary,\mathcal{H}), \mathcal{I} \times \mathcal{H} \cap \{\langle i_1, i_2\rangle | before(i_2,i_1)\} \neq \emptyset$ means "If Mary came into contact with a cat before she has a runny nose, then she has a cat allergy"

- $cronic\_pain \leftarrow$ $pain(mary, \mathcal{I}), \times(\mathcal{I}) \cap \{\langle i \rangle | duration(i) > 6\} \neq \emptyset$[10] means "If Mary has pain for more than 6 time units, then she has chronic pain"
- $flu(mary) \leftarrow$ $headache(mary, \mathcal{I}), fever(mary, \mathcal{H}), \mathcal{I} \times \mathcal{H} \cap \{\langle i_1, i_2 \rangle | during(i_1, i_2) \wedge duration(i_2) < 4\} \neq \emptyset$ means "If Mary had a headache during the interval at which she had a fever and the fever lasted for less than 4 time units, then she has the flu"

As discussed before, assumptions that need to be made to derive the head of a rule are expressed by multi-configuration pairs containing the temporal literal sequence of the rule and every configuration from the configuration set of its complete multi-configuration pair that is also in its temporal constraint. Such a multi-configuration pair is called the assumption of a rule.

DEFINITION 4.23 (ASSUMPTION OF A RULE)

Let R be a temporal rule, let $\mathcal{L}$ be its temporal literal sequence and let $\mathcal{D}_1 \cap \mathcal{D}_2 \neq \emptyset$ be its temporal equation. Multi-configuration pair $\langle \mathcal{L}, \mathcal{D}_1 \cap \mathcal{D}_2 \rangle$ is said to be the assumption of R.

To explain the definitions above more thoroughly, let R be a temporal rule and let $\mathcal{L}$ be its temporal literal sequence. Let $\mathcal{D} \cap \mathcal{B} \neq \emptyset$ be the temporal equation of R. Let $\langle \mathcal{L}, \mathcal{D} \rangle$ be a complete multi-configuration pair. $\mathcal{B}$ is then the temporal constraint of R and $\langle \mathcal{L}, \mathcal{D} \cap \mathcal{B} \rangle$ is its assumption.

$\langle \mathcal{L}, \mathcal{D} \rangle$ clearly denotes all possible situations based on the temporal literals in the body of R. In the temporal equation, $\mathcal{D}$ thus contains all possible combinations of intervals from the temporal literals in the body of the rule. Since $\mathcal{B}$ is a temporal constraint, it contains all configurations of which the elements have certain properties or relations with each other. If $\mathcal{D} \cap \mathcal{B}$ is empty it means that none of the configurations in $\mathcal{D}$ are such that their elements have these properties and relations with each other. If they were, they would be in $\mathcal{B}$. If $\mathcal{D} \cap \mathcal{B}$ is empty, this thus means that there is no situation possible on basis of the information in the body of R, in which the constraints are met. When $\mathcal{D} \cap \mathcal{B}$ is empty, the temporal equation in R is false, and the head of R may not be derived.

There are cases in which some elements of $\mathcal{D}$ are in $\mathcal{B}$, but not all. In that case $\mathcal{D} \cap \mathcal{B}$ is not empty and the temporal equation in R is true. It also means that there are configurations in $\mathcal{D}$ that meet the constraints in $\mathcal{B}$ and that there are some configurations in $\mathcal{D}$ that do not

[10] ,$\times(\mathcal{I})$ is the unary Cartesian product of $\mathcal{I}$ in prefix notation. If $\mathcal{I} = \{i_1, \dots, i_n\}$, then $\times(\mathcal{I}) = \{\langle i_1 \rangle, \dots, \langle i_n \rangle\}$.

meet these constraints. $\mathcal{D}$ is used in the temporal equation to express all possible situations based on the information in the body of R. There thus are some possible situations that meet the constraints and there are some possible situations that do not meet the constraints. Since it is not known which of the possible situations expressed by $\mathcal{D}$ is the actual situation the "real world" is in, it is not known whether in the actual situation the constraints are met or not. It would therefore be sensible to derive the head of R on basis of the assumption that the actual situation is such that it does meet the constraints. The configurations representing possible situations that do meet the constraints are in $\mathcal{D}$ as well as in $\mathcal{B}$. The assumption when deriving the head of R can therefore be denoted by the multi-configuration pair $\langle \mathcal{L}, \mathcal{D} \cap \mathcal{B} \rangle$.

Normally, multiple rules can be used to derive a literal. In the proposed logic, this does however mean that it is possible that multiple assumptions need to be made. What is assumed when deriving the head of a rule is that that the actual situation in the "real world" is one of the situations denoted by the assumption. When multiple assumptions are made to derive a literal, this literal is derived based on all those assumptions. This means that it is assumed that the actual situation in the "real world" is a situation that is denoted by all assumptions. The situations that are denoted by multiple assumptions are denoted by multi-configuration pairs that are a combination of these assumptions. The definition of such a combined multi-configuration pair is based on the definitions below.

DEFINITION 4.24 (COMBINED TEMPORAL LITERAL SEQUENCES)

Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be temporal literal sequences. $\mathcal{L}_{1,2}$ is their combined temporal literal sequence if and only if $\mathcal{L}_{1,2} = (\mathcal{L}_1 \cap \mathcal{L}_2) \cup (\overline{\mathcal{L}_1} \cap \mathcal{L}_2) \cup (\mathcal{L}_1 \cap \overline{\mathcal{L}_2})$.

Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be temporal literal sequences. Their combined temporal literal sequence $\mathcal{L}_{1,2}$ contains each element in $\mathcal{L}_1$ or in $\mathcal{L}_2$ exactly once, whether this element is in both sets or not. Below are some examples of temporal literal sequences and their combined temporal literal sequences.

EXAMPLE 4.23 (COMBINED TEMPORAL LITERAL SEQUENCES)

The following are temporal literal sequences and their combined temporal literal sequences:

- The combined temporal literal sequence of $\langle L \rangle$ and $\langle L \rangle$ is $\langle L \rangle$.
- The combined temporal literal sequence of $\langle L_1 \rangle$ and $\langle L_2 \rangle$ is $\langle L_2, L_1 \rangle$.
- The combined temporal literal sequence of $\langle L_1 \rangle$ and $\langle L_1, L_2 \rangle$ is $\langle L_1, L_2 \rangle$.
- The combined temporal literal sequence of $\langle L_1, L_2 \rangle$ and $\langle L_2, L_1 \rangle$ is $\langle L_1, L_2 \rangle$.

- The combined temporal literal sequence of $\langle L_1, L_2, L_3, L_4 \rangle$ and $\langle L_2, L_4, L_3, L_5, L_6, L_1 \rangle$ is $\langle L_2, L_4, L_3, L_5, L_6, L_1 \rangle$.

The combination of two configuration pairs denotes a group of situations which is part of the group of situations denoted by the first configuration pair, as well as part of the group of situations denoted by the second configuration pair. Two configuration pairs together can only denote such a group if the literals they have in common are true at the same intervals and if their combination does not denote a group of situations that is contradictory. If for two configuration pairs this is the case, then they can be combined.

DEFINITION 4.25 (COMBINED CONFIGURATION PAIRS)

Let $\langle \mathcal{L}_1, \mathcal{C}_1 \rangle$ and $\langle \mathcal{L}_2, \mathcal{C}_2 \rangle$ be configuration pairs. Configuration pair $\langle \mathcal{L}_{1,2}, \mathcal{C}_{1,2} \rangle$ is the combined configuration pair of $\langle \mathcal{L}_1, \mathcal{C}_1 \rangle$ and $\langle \mathcal{L}_2, \mathcal{C}_2 \rangle$, if and only if:

- $\mathcal{L}_{1,2}$ is the combined temporal literal sequence of $\mathcal{L}_1$ and $\mathcal{L}_2$, and
- If $\imath_i \in \mathcal{C}_{1,2}$, then:
  - There are $L_j \in \mathcal{L}_1$ and $\imath_j \in \mathcal{C}_1$, such that $L_j = L_i$ and $\imath_j = \imath_i$, and there are $L_k \in \mathcal{L}_2$ and $\imath_k \in \mathcal{C}_2$, such that $L_k = L_i$ and $\imath_k = \imath_i$, or
  - There is $L_j \in \mathcal{L}_1$ and $\imath_j \in \mathcal{C}_1$such that $L_j = L_i$ and $\imath_j = \imath_i$ and if $L \in \mathcal{L}_2$, then $L \neq L_i$, or
  - There is $L_j \in \mathcal{L}_2$ and $\imath_j \in \mathcal{C}_2$ such that $L_j = L_i$ and $\imath_j = \imath_i$ and if $L \in \mathcal{L}_1$, then $L \neq L_i$

Below are some examples of combined configuration pairs.

EXAMPLE 4.24 (COMBINED CONFIGURATION PAIRS)

Below are temporal literal sequences, their related configurations and their combined configurations (if they exist):

- The combined configuration pair of $\langle \langle L \rangle, \langle \imath \rangle \rangle$ and $\langle \langle L \rangle, \langle \imath \rangle \rangle$ is $\langle \langle L \rangle, \langle \imath \rangle \rangle$.
- The combined configuration pair of $\langle \langle L \rangle, \langle \imath_1 \rangle \rangle$ and $\langle \langle L \rangle, \langle \imath_2 \rangle \rangle$, where $\imath_1 \neq \imath_2$ does not exist.
- The combined configuration pair of $\langle \langle L_1 \rangle, \langle \imath_1 \rangle \rangle$ and $\langle \langle L_2 \rangle, \langle \imath_2 \rangle \rangle$ is $\langle \langle L_2, L_1 \rangle, \langle \imath_2, \imath_1 \rangle \rangle$.
- The combined configuration pair of $\langle \langle L_1 \rangle, \langle \imath_1 \rangle \rangle$ and $\langle \langle L_1, L_2 \rangle, \langle \imath_1, \imath_2 \rangle \rangle$ is $\langle \langle L_1, L_2 \rangle, \langle \imath_1, \imath_2 \rangle \rangle$.
- The combined configuration pair of $\langle \langle L_1 \rangle, \langle \imath_1 \rangle \rangle$ and $\langle \langle L_1, L_2 \rangle, \langle \imath_3, \imath_2 \rangle \rangle$, where $\imath_3 \neq \imath_1$, does not exist.

- The combined configuration pair of $\langle\langle L_1, L_2\rangle, \langle i_1, i_2\rangle\rangle$ and $\langle\langle L_2, L_1\rangle, \langle i_2, i_1\rangle\rangle$ is $\langle\langle L_1, L_2\rangle, \langle i_1, i_2\rangle\rangle$.
- The combined configuration pair of $\langle\langle L_1, L_2\rangle, \langle i_1, i_2\rangle\rangle$ and $\langle\langle L_2, L_1\rangle, \langle i_3, i_1\rangle\rangle$, where $i_3 \neq i_2$, does not exist.
- The combined configuration pair of $\langle\langle L_1, L_2, L_3, L_4\rangle, \langle i_1, i_2, i_3, i_4\rangle\rangle$ and $\langle\langle L_2, L_4, L_3, L_5, L_6\rangle, \langle i_2, i_4, i_5, i_3, i_6\rangle\rangle$ is $\langle\langle L_2, L_4, L_3, L_5, L_6, L_1\rangle, \langle i_2, i_4, i_3, i_5, i_6, i_1\rangle\rangle$.
- The combined configuration pair of $\langle\langle L_1, L_2, L_3, L_4\rangle, \langle i_1, i_2, i_3, i_4\rangle\rangle$ and $\langle\langle L_2, L_4, L_3, L_5, L_6\rangle, \langle i_3, i_4, i_5, i_2, i_6\rangle\rangle$, where $i_3 \neq i_2$ does not exist.

In a configuration pair, implicitly each element of the configuration is bound by the literal in the temporal literal sequence that has the same position in the sequence. This property arises directly from definition 4.15. If $\mathcal{L} = \langle L_1, \ldots, L_n\rangle$ and $\mathcal{C} = \langle i_1, \ldots, i_n\rangle$ form a configuration pair $\langle\mathcal{L}, \mathcal{C}\rangle$, then we can say for each $i_i \in \mathcal{C}$ that it is bound by $L_i \in \mathcal{L}$.

Let $\langle\mathcal{L}_1, \mathcal{C}_1\rangle$ and $\langle\mathcal{L}_2, \mathcal{C}_2\rangle$ be configuration pairs. Their combined configuration pair $\langle\mathcal{L}_{1,2}, \mathcal{C}_{1,2}\rangle$ (if it exists) contains the combined temporal literal sequence of $\mathcal{L}_1$ and $\mathcal{L}_2$, viz. $\mathcal{L}_{1,2}$. The configuration of this pair is a combination of configurations $\mathcal{C}_1$ and $\mathcal{C}_2$, viz. $\mathcal{C}_{1,2}$.

According to the definition above, each element of $\mathcal{C}_{1,2}$ is:

- an element of both $\mathcal{C}_1$ and $\mathcal{C}_2$ that is bound by a literal with is in $\mathcal{L}_1$ as well as $\mathcal{L}_2$
- an element of $\mathcal{C}_1$ that is bound by a literal that is not in $\mathcal{L}_2$, or
- an element of $\mathcal{C}_2$ that is bound by a literal that is not in $\mathcal{L}_1$, or

If there is a literal in $\mathcal{L}_1$ that is also in $\mathcal{L}_2$, but it does not bind the same element in $\mathcal{C}_1$ as in $\mathcal{C}_2$, then by definition it is impossible to combine $\langle\mathcal{L}_1, \mathcal{C}_1\rangle$ and $\langle\mathcal{L}_2, \mathcal{C}_2\rangle$, and $\langle\mathcal{L}_{1,2}, \mathcal{C}_{1,2}\rangle$ does thus not exist.

From a semantic point of view the above makes sense as well. Recall that configuration pairs denote a group of situations in the “real world” that are possible according to their temporal literal sequences. A combined configuration pair of two configuration pairs denotes a group of possible situations in which all literals that are in the temporal literal sequence of one or both originating configuration pairs are true at the same intervals as denoted by those pairs. In other words, a combined configuration pair denotes a group of situations which is part of the groups of situations denoted by their originating configuration pairs. A group of situations can only be part of two other groups of situations if in these groups the same literals are true at the same intervals. If it is the case that according to two configuration pairs the exact same literal containing the same interval set is true at different times, then there is no group of situations that is denoted by both configuration pairs. Note that by the definition above, these are exactly the same cases in which two configuration pairs do not have a combined configuration pair. Note in addition that situations in which the same literal is true and false at overlying intervals are not considered to be possible and according to definition 4.15 cannot be described by a configuration pair and thus not by a combined configuration pair.

Multi-configuration pairs such as the ones denoting assumptions can also be combined.

DEFINITION 4.26 (COMBINED MULTI-CONFIGURATION PAIRS)

Let $\langle \mathcal{L}_1, \mathcal{D}_1 \rangle$ and $\langle \mathcal{L}_2, \mathcal{D}_2 \rangle$ be multi-configuration pairs. Multi-configuration pair $\langle \mathcal{L}_{1,2}, \mathcal{D}_{1,2} \rangle$ is the combined multi-configuration pair of $\langle \mathcal{L}_1, \mathcal{D}_1 \rangle$ and $\langle \mathcal{L}_2, \mathcal{D}_2 \rangle$, if and only if:

- $\mathcal{L}_{1,2}$ is the combined temporal literal sequence of $\mathcal{L}_1$ and $\mathcal{L}_2$, and
- If $\langle \mathcal{L}_{1,2}, \mathcal{C} \rangle$ is the combined configuration pair of $\langle \mathcal{L}_1, \mathcal{C}_i \rangle$ and $\langle \mathcal{L}_2, \mathcal{C}_j \rangle$, where $\mathcal{C}_i \in \mathcal{D}_1$ and $\mathcal{C}_j \in \mathcal{D}_2$, then $\mathcal{C} \in \mathcal{D}_{1,2}$, and
- If $\mathcal{C} \in \mathcal{D}_{1,2}$, then there are $\mathcal{C}_i \in \mathcal{D}_1$ and $\mathcal{C}_j \in \mathcal{D}_2$, such that $\langle \mathcal{L}_{1,2}, \mathcal{C} \rangle$ is the combined configuration pair of $\langle \mathcal{L}_1, \mathcal{C}_i \rangle$ and $\langle \mathcal{L}_2, \mathcal{C}_j \rangle$

REMARKS

The combined multi-configuration pair of multi-configuration pairs $\langle \mathcal{L}_1, \mathcal{D}_1 \rangle \ldots \langle \mathcal{L}_n, \mathcal{D}_n \rangle$ is the multi-configuration pair $\langle \mathcal{L}_{1,\ldots,n}, \mathcal{D}_{1,\ldots,n} \rangle$, which is obtained by combining $\langle \mathcal{L}_1, \mathcal{D}_1 \rangle$ with $\langle \mathcal{L}_2, \mathcal{D}_2 \rangle$ and combining the thus obtained multi-configuration pair with $\langle \mathcal{L}_3, \mathcal{D}_3 \rangle$ and combining the thus obtained multi-configuration pair with … and combining the thus obtained multi-configuration pair with $\langle \mathcal{L}_n, \mathcal{D}_n \rangle$.

Below are some examples of combined multi-configuration pairs.

EXAMPLE 4.25 (COMBINED MULTI-CONFIGURATION PAIRS)

The following are multi-configuration pairs and their combined multi-configuration pairs:

- The combined multi-configuration pair of $\langle \langle L \rangle, \{\langle \imath \rangle\} \rangle$ and $\langle \langle L \rangle, \{\langle \imath \rangle\} \rangle$ is $\langle \langle L \rangle, \{\langle \imath \rangle\} \rangle$.
- The combined multi-configuration pair of $\langle \langle L \rangle, \{\langle \imath_1 \rangle, \langle \imath_2 \rangle\} \rangle$ and $\langle \langle L \rangle, \{\langle \imath_1 \rangle, \langle \imath_2 \rangle\} \rangle$ is $\langle \langle L \rangle, \{\langle \imath_1 \rangle, \langle \imath_2 \rangle\} \rangle$.
- The combined multi-configuration pair of $\langle \langle L \rangle, \{\langle \imath_1 \rangle, \langle \imath_2 \rangle\} \rangle$ and $\langle \langle L \rangle, \{\langle \imath_1 \rangle, \langle \imath_3 \rangle\} \rangle$, where $\imath_2 \neq \imath_3$ is $\langle \langle L \rangle, \{\langle \imath_1 \rangle\} \rangle$.

- $\langle\langle L_1\rangle,\{\langle i_1\rangle,\langle i_2\rangle\}\rangle$ and $\langle\langle L_2\rangle,\{\langle i_3\rangle,\langle i_4\rangle\}\rangle$, is $\langle\langle L_1,L_2\rangle,\left\{\begin{matrix}\langle i_1,i_3\rangle,\langle i_1,i_4\rangle,\\ \langle i_2,i_3\rangle,\langle i_2,i_4\rangle\end{matrix}\right\}\rangle$.
- The combined multi-configuration pair of $\langle\langle L_1,L_2\rangle,\{\langle i_1,i_2\rangle,\langle i_3,i_4\rangle\}\rangle$ and $\langle\langle L_1,L_3\rangle,\{\langle i_1,i_5\rangle,\langle i_6,i_7\rangle\}\rangle$, is $\langle\langle L_1,L_2,L_3\rangle,\{\langle i_1,i_2,i_5\rangle\}\rangle$.
- The combined multi-configuration pair of $\langle\langle L_1,L_2\rangle,\{\langle i_1,i_2\rangle,\langle i_3,i_4\rangle\}\rangle$ and $\langle\langle L_1,L_3\rangle,\{\langle i_1,i_5\rangle,\langle i_3,i_7\rangle\}\rangle$, is $\langle\langle L_1,L_2,L_3\rangle,\left\{\begin{matrix}\langle i_1,i_2,i_5\rangle,\\ \langle i_3,i_4,i_7\rangle\end{matrix}\right\}\rangle$.
- The combined multi-configuration pair of $\langle\langle L_1,L_2\rangle,\{\langle i_1,i_2\rangle,\langle i_3,i_4\rangle\}\rangle$ and $\langle\langle L_1,L_3\rangle,\{\langle i_5,i_6\rangle,\langle i_7,i_8\rangle\}\rangle$, is $\langle\langle L_1,L_2,L_3\rangle,\emptyset\rangle$.

A multi-configuration pair denotes several groups of situations that are possible according to its temporal literal sequence. The combination of two or more multi-configuration pairs denotes all groups of situations that are part of a group of situations denoted by each of its originating multi-configuration pairs.

Multi-configuration pairs are compatible if there is at least one group of situations that is part of the groups of situations denoted by every one of them. If such a group of situations does not exist, then their combined multi-configuration pair does not denote any situation. Its configuration set should then be empty, since the number of groups of situations described by a multi-configuration pair is the same as the number of configurations in its configuration set.

DEFINITION 4.27 (COMPATIBLE MULTI-CONFIGURATION PAIRS)

Let $\langle\mathcal{L}_1,\mathcal{D}_1\rangle$ … $\langle\mathcal{L}_n,\mathcal{D}_n\rangle$ be multi-configuration pairs. Let $\langle\mathcal{L}_{1,\dots,n},\mathcal{D}_{1,\dots,n}\rangle$ be their combined multi-configuration pair. $\langle\mathcal{L}_1,\mathcal{D}_1\rangle$ … $\langle\mathcal{L}_n,\mathcal{D}_n\rangle$ are said to be compatible if and only if $\mathcal{D}_{1,\dots,n}$ is not empty.

Below are some examples of compatible and incompatible multi-configuration pairs.

EXAMPLE 4.26 (COMPATIBLE MULTI-CONFIGURATION PAIRS)

The following multi-configuration pairs are compatible:

- $\langle\langle L\rangle,\{\langle i\rangle\}\rangle$ and $\langle\langle L\rangle,\{\langle i\rangle\}\rangle$
- $\langle\langle L\rangle,\{\langle i_1\rangle,\langle i_2\rangle\}\rangle$ and $\langle\langle L\rangle,\{\langle i_1\rangle,\langle i_2\rangle\}\rangle$
- $\langle\langle L\rangle,\{\langle i_1\rangle,\langle i_2\rangle\}\rangle$ and $\langle\langle L\rangle,\{\langle i_1\rangle,\langle i_3\rangle\}\rangle$,
- $\langle\langle L_1\rangle,\{\langle i_1\rangle,\langle i_2\rangle\}\rangle$ and $\langle\langle L_2\rangle,\{\langle i_3\rangle,\langle i_4\rangle\}\rangle$
- $\langle\langle L_1,L_2\rangle,\{\langle i_1,i_2\rangle,\langle i_3,i_4\rangle\}\rangle$ and $\langle\langle L_1,L_3\rangle,\{\langle i_1,i_5\rangle,\langle i_6,i_7\rangle\}\rangle$

The following multi-configuration pairs are not compatible:

- $\langle\langle L\rangle, \{\langle \imath_1\rangle\}\rangle$ and $\langle\langle L\rangle, \{\langle \imath_2\rangle\}\rangle$, where $\imath_1 \neq \imath_2$
- $\langle\langle L\rangle, \{\langle \imath_1\rangle, \langle \imath_2\rangle\}\rangle$ and $\langle\langle L\rangle, \{\langle \imath_3\rangle, \langle \imath_4\rangle\}\rangle$, where $\imath_1 \neq \imath_3$ and $\imath_1 \neq \imath_4$ and $\imath_2 \neq \imath_3$ and $\imath_2 \neq \imath_4$
- $\langle\langle L_1, L_2\rangle, \{\langle \imath_1, \imath_2\rangle, \langle \imath_3, \imath_4\rangle\}\rangle$ and $\langle\langle L_1, L_3\rangle, \{\{\langle \imath_5, \imath_6\rangle, \langle \imath_7, \imath_8\rangle\}\}\rangle$, where $\imath_1 \neq \imath_5$ and $\imath_1 \neq \imath_7$ and $\imath_3 \neq \imath_5$ and $\imath_3 \neq \imath_7$

Let R be a temporal rule with temporal literal sequence $\mathcal{L}$ and temporal equation $\mathcal{D}_1 \cap \mathcal{D}_2 \neq \emptyset$. Its assumption is then $\langle \mathcal{L}, \mathcal{D}_1 \cap \mathcal{D}_2\rangle$. As discussed before, it is important that it is possible that the assumptions in a derivation describe the same situations, as it is assumed that the actual situation is one of them. It would for instance be nonsensical to make an assumption that “Mary had a headache for more than 3 hours” and that “Mary had a headache for less than 3 hours” in the same derivation[11]. To check that there are situations denoted by all assumptions, it may be determined whether the multi-configuration pairs denoting them are compatible.

Consider the following multi-configuration pairs denoting the assumption in the example above:

1. $\langle\langle headache(mary, \mathcal{I})\rangle, \times(\mathcal{I}) \cap \{\langle \imath\rangle | duration(\imath) > 3\}\rangle$, and
2. $\langle\langle headache(mary, \mathcal{I})\rangle, \times(\mathcal{I}) \cap \{\langle \imath\rangle | duration(\imath) < 3\}\rangle$

Clearly the configuration sets of 1 and 2 are disjoint. The fact that they are disjoint obviously arises from the fact that 1 and 2 cannot denote same situations. It is impossible for $\mathcal{I}$ to contain an interval that has a duration that is more as well as less than 3 time units. The combined multi-configuration pair of 1 and 2 will have an empty configuration set, since there are no configurations in the configuration sets of both 1 and 2. The fact that the configuration set of the combination of 1 and 2 is empty does directly arise from the fact that they cannot possibly describe the same situations. Since the configuration set of the combination of 1 and 2 is empty, 1 and 2 are incompatible.

Since we are able to determine what assumptions exactly may need to be made in a derivation and we are able to determine whether it is possible to make multiple assumptions in a derivation, it is possible to define the notion of a defeasible derivation in the proposed logic. Assumptions are part of a defeasible derivation and assumptions that are incompatible, as defined above cannot be made in the same derivation. Including assumptions in derivations makes it possible to make it explicit on which assumptions they are based. This will be of great importance later in the dialectical process.

The definition of a defeasible derivation in the proposed logic is similar to the definition in (García & Simari, 2004). The only difference is that when temporal rules are used, their

[11] Providing that the assumption is made on the same fact/literal, i.e. the assumption is about the same time Mary had a headache.

assumptions are part of the derivation and the assumptions in the derivation should all be compatible. This makes sure that assumptions made in the derivation are consistent.

DEFINITION 4.28 (DEFEASIBLE DERIVATIONS)

Let $\mathcal{P} = (\Pi, \Delta)$ be a de.l.p.. A defeasible derivation of a literal L from $\mathcal{P}$, is a finite sequence $L_1, L_2, \dots, L_n, L$ of ground literals and assumptions and each L_i is in the sequence because:

- L_i is a fact in Π, or
- There exists a rule R_i in Π or Δ with head L_i and body $\langle B_1, \dots, B_k \rangle$, and
    - Every $B_j \in \langle B_1, \dots, B_k \rangle$ is a true temporal equation or an element L_j of the sequence appearing before L_i $(j < i)$, and
    - If R_i is a temporal rule, then its assumption is an element L_j of the sequence appearing before L_i, $(j < i)$, and
    - If R_i is a temporal rule, then its assumption is compatible with the combined assumptions of all assumptions prior in the sequence

A multi-configuration pair is called the assumption of the derivation of a literal L if and only if it is the combined multi-configuration pair of all assumptions in the derivation of L.

Below is an example of a defeasible derivation.

EXAMPLE 4.27 (DEFEASIBLE DERIVATIONS)

Consider the following de.l.p. $\mathcal{P}_{4.27} = (\Pi, \Delta)$, with
$\Pi =$
$$\left\{ \begin{array}{c} P(x, \mathcal{I}) \leftarrow Q(x, \mathcal{I}), \\ Q(x, \mathcal{I}) \leftarrow R(x, \mathcal{I}), \times (\mathcal{I}) \cap \{\langle i \rangle | duration(i) \geq 2\} \neq \emptyset, \\ R(x, \mathcal{I}) \leftarrow S(x, \mathcal{I}), T(x, \mathcal{H}), \mathcal{I} \times \mathcal{H} \cap \{\langle i_1, i_2 \rangle | before(i_1, i_2)\} \neq \emptyset \\ T(c, \{(9,10), (19,28)\}), \\ S(c, \{(1,3), (5,8)\}), \\ T(b, \{(4,7), (12,15)\}), \\ S(b, \{(1,3), (5,8)\}) \end{array} \right\}$$
and $\Delta = \emptyset$.

There is a defeasible derivation for $P(b, \{(1,3), (5,8)\})$ from $\mathcal{P}_{4.27}$, viz.

$$\left\langle \begin{matrix} S(b,\{(1,3),(5,8)\},T(b,\{(4,7),(12,15)\}), \\ \left\langle \begin{matrix} \langle S(b,\{(1,3),(5,8)\},T(b,\{(4,7),(12,15)\})\rangle, \\ \{(1,3),(5,8)\} \times \{(4,7),(12,15)\} \cap \{\langle i_1,i_2\rangle | before(i_1,i_2)\} \end{matrix} \right\rangle \\ \langle\langle R(b,\{(1,3),(5,8)\})\rangle, \times (\{(1,3),(5,8)\}) \cap \{\langle i\rangle | duration(i) \geq 2\}\rangle \\ R(b,\{(1,3),(5,8)\}), Q(b,\{(1,3),(5,8)\}), P(b,\{(1,3),(5,8)\}) \end{matrix} \right\rangle$$

.

In (García & Simari, 2004) strict derivations are derivations in which only facts and strict rules are used, arguments based on strict derivations cannot be defeated. In the proposed logic however, a literal that is derived on basis of assumptions may be false, since the actual situation may be one of the situations that is not denoted by the assumption. In other words, assumptions may be false. Therefore, a literal derived on basis of only strict rules and facts and assumptions could also be false and should be defeasible, even though no defeasible rules have been used. This possibility more specifically arises in case the combined multi-configuration pair of all assumptions in the derivation does not denote all possible situations on basis of its temporal literal sequence. Such a combined multi-configuration denotes the assumption that the actual situation is one of the situations denoted by it and not one of the other situations that are possible according to its temporal literal sequence. The actual situation in the "real world" could however very well be any of these other possible situations. Since this is a possibility, it may also be possible that the literal that is derived is false. Therefore, in the proposed logic derivations are only considered to be strict if only strict rules and facts are used and in addition, if the combined multi-configuration set of all assumptions in the derivation is complete.

DEFINITION 4.29 (STRICT DERIVATIONS)

Let $\mathcal{P} = (\Pi, \Delta)$ be a de.l.p.. Let h be a literal with a defeasible derivation $L_1, L_2, \dots, L_n, L$ from $\mathcal{P}$. We will say that L has a strict derivation from $\mathcal{P}$, if and only if:

- either L is a fact in Π, or all rules used for obtaining the sequence $L_1, L_2, \dots, L_n$ are strict rules, and
- if $L_1, L_2, \dots, L_n$ contains assumptions, then the combined multi-configuration pair of all assumptions in $L_1, L_2, \dots, L_n$ is complete.

Below is an example of a strict derivation.

EXAMPLE 4.28 (STRICT DERIVATIONS)

Consider de.l.p. $\mathcal{P}_{4.27}$ from example 4.27 once again.

There is a strict derivation for $P(c,\{(1,3),(5,8)\})$ from $\mathcal{P}_{4.27}$, viz.

$$\left\langle \begin{matrix} S(c,\{(1,3),(5,8)\},T(c,\{(9,10),(19,28)\}), \\ R(c,\{(1,3),(5,8)\}), Q(c,\{(1,3),(5,8)\}), P(c,\{(1,3),(5,8)\}) \end{matrix} \right\rangle.$$

Literal $P(b, \{(1,3),(5,8)\})$ cannot be derived strictly from $\mathcal{P}_{4.27}$, since the third rule in Π has to be used to derive $R(b, \{(1,3),(5,8)\})$ and an assumption has to be made to do this. This assumption is not a complete multi-configuration pair.

In the next section, argument structures are defined on basis of the definitions of strict and defeasible derivations above. Attack and defeat between arguments are defined as well.

4.4 ARGUMENT STRUCTURES, ATTACK AND DEFEAT

Similar to (García & Simari, 2004), in the proposed logic, the notion of a contradictory set of rules is used in the definition of an argument structure. The definition of a contradictory set of rules is different from the definition in (García & Simari, 2004). Complementary literals however are defined in a similar manner.

DEFINITION 4.30 (COMPLEMENTARY LITERALS)

Let L_1 and L_2 be literals. L_1 and L_2 are said to be complementary if and only if $L_1 = \neg L_2$ or if $L_2 = \neg L_1$.

Below are some examples of complementary literals.

EXAMPLE 4.29 (COMPLEMENTARY LITERALS)

The following literals are complementary:

- $P(c)$ and $\neg P(c)$
- $P(c, \mathcal{I})$ and $\neg P(c, \mathcal{I})$
- $fever(mary, \{(1,3),(4,7)\})$ and $\neg fever(mary, \{(1,3),(4,7)\})$

The following literals are not complementary:

- $P(c_1)$ and $\neg P(c_2)$, where $c_1 \neq c_2$
- $P(c, \mathcal{I})$ and $\neg P(c, \mathcal{H})$, where $\mathcal{I} \neq \mathcal{H}$
- $fever(mary, \{(1,3),(4,7)\})$ and $\neg headache(mary, \{(1,3),(4,7)\})$

In the proposed logic there are two kinds of literals, temporal and non-temporal ones. In (García & Simari, 2004) a set of rules is contradictory if two complementary literals can be derived. For non-temporal literals this definition remains valid. Temporal literals however do not necessarily contradict each other if they are complements. Normally a contradiction arises when it can be derived that something is true as well as false. Temporal literals take temporal information into account. When temporal information is taken into account, there is a contradiction if something is true and false at the same time. A temporal literal containing a temporal term denoting an interval set with multiple intervals expresses that it can be true at any of the intervals in its temporal term. This means that if there are two temporal literals that are complements, one of the groups of situations they denote may not be contradictory.

As an example, let $P(c_1, c_2, \{i_1, i_2\})$ and $\neg P(c, \{i_1, i_2\})$ be temporal literals, where i_1 and i_2 are not overlying. Clearly these literals are complementary. On basis of the information in these literals, the following four groups of situations can be described:

1. The relation P between c_1 and c_2 holds at interval i_1 and does not hold at interval i_1
2. The relation P between c_1 and c_2 holds at interval i_1 and does not hold at interval i_2
3. The relation P between c_1 and c_2 holds at interval i_2 and does not hold at interval i_1
4. The relation P between c_1 and c_2 holds at interval i_2 and does not hold at interval i_2

Clearly the situations described in 1 and 4 are contradictory and thus are not possible. The situations described in 2 and 3 clearly are possible on basis of the information in the complementary literals. $P(c_1, c_2, \{i_1, i_2\})$ and $\neg P(c_1, c_2, \{i_1, i_2\})$ thus do not only describe situations that are contradictory and a set of rules deriving them should not be considered contradictory as long as no other contradictions can be derived from them. Note that the multi-configuration set $\langle\langle P(c_1, c_2, \{i_1, i_2\}), \neg P(c_1, c_2, \{i_1, i_2\})\rangle, \{\langle i_1, i_2\rangle, \langle i_2, i_1\rangle\}\rangle$ is complete.

Two temporal literals are contradictory if all situations they denote are contradictory. In that case, there are no possible situations on basis of the information in those temporal literals. This means that if two temporal literals are contradictory, then any multi-configuration pair containing these temporal literals in their temporal literal sequence has an empty configuration set, even if they are complete.

As an example, let $P(c_1, c_2, \{i_1\})$ and $\neg P(c_1, c_2, \{i_1, i_2\})$ be temporal literals, where i_1 overlies with i_2. The following groups of situations can be described:

1. The relation P between c_1 and c_2 holds at interval i_1 and does not hold at interval i_1
2. The relation P between c_1 and c_2 holds at interval i_1 and does not hold at interval i_2

Clearly both groups contain only situations that are contradictory and thus not possible. It follows from definitions 4.15, 4.17 and 4.18 that multi-configuration pair $\langle\langle P(c_1, c_2, \{i_1\}), \neg P(c_1, c_2, \{i_1, i_2\})\rangle, \emptyset\rangle$ is complete. The fact that the configuration set is empty follows directly from the fact that all situations that can be described on basis of the information in it are contradictory.

In some cases, a set of more than two temporal literals is contradictory, while any subset of this set is not. As an example, let $P(c_1, c_2, \{i_1, i_2\})$, $\neg P(c_1, c_2, \{i_1\})$ and $\neg P(c_1, c_2, \{i_2\})$ be temporal literals. Let i_1 and i_2 not be overlying. The following groups of situations can be described:

1. The relation P between c_1 and c_2 holds at interval i_1 and does not hold at interval i_1 and interval i_2
2. The relation P between c_1 and c_2 holds at interval i_2 and does not hold at interval i_1 and interval i_2

Clearly all situations described above are contradictory and multi-configuration pair $\langle\langle P(c_1, c_2, \{i_1, i_2\}), \neg P(c_1, c_2, \{i_1\}), \neg P(c_1, c_2, \{i_2\})\rangle, \emptyset\rangle$ is complete. Any complete multi-configuration pair containing a subset of the temporal literal sequence in the multi-configuration pair above contains a non-empty configuration set. It may therefore be concluded that contradiction may arise from more than two temporal literals.

The definition of a contradictory set of temporal literals is below.

DEFINITION 4.31 (CONTRADICTORY SETS OF LITERALS)

A set of literals $\Gamma = \{L_1, \dots, L_n\}$ is contradictory if and only if:

- There are two non-temporal literals $L_i, L_j \in \Gamma$ and L_i and L_j are complements, or
- Γ contains a subset of temporal literals $\{L_i, \dots, L_k\}$ and multi-configuration pair $\langle\langle L_i, \dots, L_k\rangle, \emptyset\rangle$ is complete.

Below are some examples of sets of literals that are contradictory.

EXAMPLE 4.30 CONTRADICTORY SETS LITERALS

The following sets of literals are contradictory:

- $\{P(c), \neg P(c)\}$
- $\{P(c, \{i\}), \neg P(c, \{i\}), Q(c)\}$
- $\{P(c, \{i_1, i_2\}), \neg P(c, \{i_1, i_2\})\}$, where i_1 and i_2 overlie
- $\{P(c, \{i_1, i_2\}), P(c, \{i_3\}), \neg P(c, \{i_1, i_3\}), \neg P(c, \{i_2\})\}$

The following sets of temporal literals are not contradictory:

- $\{P(c_1, \{i\}), \neg P(c_2, \{i\})\}$, where $c_1 \neq c_2$
- $\{P(c, \{i_1, i_2\}), \neg P(c, \{i_1, i_2\})\}$, where i_1 and i_2 not overlie
- $\{P(c, \{i_1, i_2\}), P(c, \{i_2\}), \neg P(c, \{i_1, i_3\}), \neg P(c, \{i_2\})\}$

A set of rules can now be defined as contradictory if the set of literals derived from it is contradictory.

DEFINITION 4.32 (CONTRADICTORY SETS OF RULES)

A set of rules and facts is contradictory if and only if there exists a defeasible derivation for a set of literals from it that is contradictory.

Below is an example of a contradictory set of rules.

EXAMPLE 4.31 (CONTRADICTORY SETS OF RULES)

Consider the following set of facts and rules:

$$\left\{\begin{array}{c} P(x,\mathcal{I}) \prec Q(x,\mathcal{I}), \\ P(x,\mathcal{I}) \prec \neg R(x,\mathcal{I}), \\ \neg P(x,\mathcal{I}) \prec S(x,\mathcal{I}), \\ \neg P(x,\mathcal{I}) \prec T(x,\mathcal{I}), \\ Q(c,\{(1,5),(6,9)\}), \\ \neg R(c,\{(7,9)\}), \\ S(c,\{(2,3),(21,30)\}), \\ T(c,\{(4,7)\}) \end{array}\right\}$$

From this set the literals $P(c,\{(1,5),(6,9)\})$, $P(c,\{(7,9)\})$, $\neg P(c,\{(2,3),(21,30)\})$ and $\neg P(c,\{(4,7)\})$ may be derived. The complete multi-configuration pair of a temporal literal sequence containing these literals has an empty configuration set. The set of rules and facts above is thus contradictory.

Argument structures in the proposed logic are defined similar to the argument structures in (García & Simari, 2004). There are however some differences. While argument structures in DeLP contain a set with only defeasible rules, argument structures in the proposed logic contain a set with defeasible as well as strict rules. This is necessary since defeasibility in the proposed logic does not only arise from the use of defeasible rules, but also from the use of interval sets and assumptions.

In (García & Simari, 2004), there is a convention that the set of strict rules and facts in a de.l.p. is not contradictory, since only indefeasible literals may be derived from this set. In the proposed logic this would not make a lot of sense since it is possible to make assumptions when using strict rules. The replacing convention in the proposed logic will thus be that the set of literals that have a strict derivation from a de.l.p. is not contradictory. In (García & Simari, 2004), the union of the set of rules in an argument structure and the set of strict rules and facts is not allowed to be contradictory, to prevent the creation of argument structures that are self-defeating. This requirement is adopted in the proposed logic in a form that is adapted according to the new convention.

Argument structures are used to attack and defeat each other. In (García & Simari, 2004), an argument structure can attack another argument structure if their conclusions are complements. As shown before, in the proposed logic contradiction may arise from literals that are not complements or from more than two literals. Attack is in principle a binary relation and to not unnecessarily complicate the argumentation process, it should be kept that way. The way in which the new notion of contradiction between literals is incorporated in the proposed logic is by defining argument structures to have a set of literals as their conclusion instead of a single literal. In that way, most of the argumentation process defined by (García & Simari, 2004) stays intact, while attack between argument structures still is only possible if their conclusions contradict each other. In addition, it provides the possibility to put restrictions on the sets of argument structures that attack each other. One of the restrictions may be that the assumptions on which they are based need to be compatible (the third requirement in the definition below). If needed, these restrictions can be easily adapted by adding or removing requirements from definition 4.33.

DEFINITION 4.33 (ARGUMENT STRUCTURES)

Let $\Gamma = \{h_1, \dots, h_n\}$ be a non-empty set of literals. Let $\mathcal{P} = (\Pi, \Delta)$ be a de.l.p.. Let the subset of facts in Π be denoted by Φ. Let the set of literals that have a strict derivation from $\mathcal{P}$ be denoted by Σ. We say that $\langle \mathcal{A}, \Gamma \rangle$ is an argument structure for the literals in Γ, if $\mathcal{A}$ is a set of defeasible and strict rules of Δ and Π, such that:

1. There exists a defeasible or strict derivation for each of the literals in Γ from $\Phi \cup \mathcal{A}$, and
2. The set $\Sigma \cup \mathcal{A}$ is non-contradictory, and
3. The assumptions of the temporal rules in $\mathcal{A}$ are compatible, and
4. $\mathcal{A}$ is minimal: there is no proper subset $\mathcal{A}'$ of $\mathcal{A}$ such that the literals in Γ can be derived from $\mathcal{A}'$ and $\mathcal{A}'$ satisfies conditions 1., 2. and 3..

REMARKS

The conclusion of an argument structure does not need to contain all literals that can be derived by the rules in its argument as long as 1., 2., 3. and 4. are satisfied and the conclusion is not empty.

EXAMPLE 4.32 (ARGUMENT STRUCTURES)

Let $\mathcal{P}_{4.32} = (\Pi, \Delta)$ be a de.l.p. with $\Pi = \left\{ \begin{matrix} Q(x,\mathcal{I}) \leftarrow S(x,\mathcal{I}), \\ \neg P(x,\mathcal{I}) \leftarrow S(x,\mathcal{I}), \\ S(c,\{(1,5),(4,7)\}) \end{matrix} \right\}$
and
$\Delta = \{P(x,\mathcal{I}) -\!\!\prec Q(x,\mathcal{I}), \times(\mathcal{I}) \cap \{\langle i \rangle | duration(i) < 5\} \neq \emptyset\}$.

There are strict derivations for literals $S(c,\{(1,5),(4,7)\})$ and $R(c,\{(10,15)\})$ from $\mathcal{P}_{4.32}$. Their argument structures are

$\langle \emptyset, \{S(c, \{(1,5), (4,7)\})\}\rangle$ and $\langle \emptyset, \{R(c, \{(10,15)\})\}\rangle$ respectively.

There is in addition a strict derivation for $Q(c, \{(1,5), (4,7)\})$, its argument structure is
$\left\langle \begin{matrix} \{Q(c, \{(1,5), (4,7)\}) \leftarrow S(c, \{(1,5), (4,7)\})\}, \\ \{Q(c, \{(1,5), (4,7)\})\} \end{matrix} \right\rangle$.

There is also a strict derivation for $\neg P(c, \{(1,5), (4,7)\})$ from $\mathcal{P}_{4.32}$. Its argument structure is
$\left\langle \begin{matrix} \{\neg P(c, \{(1,5), (4,7)\}) \leftarrow S(c, \{(1,5), (4,7)\})\}, \\ \{\neg P(c, \{(1,5), (4,7)\})\} \end{matrix} \right\rangle$.

There is a defeasible derivation for $P(c, \{(1,5), (4,7)\})$. There is however no argument structure for this literal, since a set containing this literal and the literals that are derived strictly is contradictory.

The following are additional argument structures:

- $\langle \emptyset, \{S(c, \{(1,5), (4,7)\}), \{R(c, \{(10,15)\})\}\}\rangle$
- $\left\langle \begin{matrix} \{Q(c, \{(1,5), (4,7)\}) \leftarrow S(c, \{(1,5), (4,7)\})\}, \\ \{Q(c, \{(1,5), (4,7)\}), R(c, \{(10,15)\})\} \end{matrix} \right\rangle$

Subargument structures are defined the similar as in (García & Simari, 2004).

DEFINITION 4.34 (SUBARGUMENT STRUCTURE)

An argument structure $\langle \mathcal{B}, \Gamma_1\rangle$ is a subargument structure of $\langle \mathcal{A}, \Gamma_2\rangle$ if $\mathcal{B} \subseteq \mathcal{A}$.

EXAMPLE 4.33 (SUBARGUMENT STRUCTURES)

Consider the argument structure for literal $Q(c, \{(1,5), (4,7)\})$ in example 4.32 once again. The argument structure for $S(c, \{(1,5), (4,7)\})$ is a subargument structure of it.

Disagreement in the proposed logic is between two sets of literals instead of between two literals. In addition, in the definition below, the set of strict rules and facts has been replaced by the set of literals that have a strict derivation. This has to do with the changed convention that the set of literals that have a strict derivation is not contradictory instead of the set of strict rules and facts. The definition is otherwise equivalent to the corresponding definition in (García & Simari, 2004).

DEFINITION 4.35 (DISAGREEMENT)

Let $\mathcal{P} = (\Pi, \Delta)$ be a de.l.p.. Let Σ be the set of all literals that have a strict derivation from $\mathcal{P}$. A non-empty set of literals Γ_1 and a non-empty set of literals Γ_2 disagree if and only if the set $\Sigma \cup \Gamma_1 \cup \Gamma_2$ is contradictory.

REMARKS

An extra requirement to rule out cases in which one of the disagreeing sets by itself is contradictory is not necessary, since disagreement is used to define which arguments attack each other and the conclusions of arguments cannot be contradictory according to the second requirement in definition 4.33.

Counterarguments are defined similar as in (García & Simari, 2004). The only difference is that argument structures have a set of literals as their conclusion.

DEFINITION 4.36 (COUNTERARGUMENTS)

We say that $\langle \mathcal{A}_1, \Gamma_1 \rangle$ counterargues, rebuts or attacks $\langle \mathcal{A}_2, \Gamma_2 \rangle$ at the literals in Γ, if and only if there exists a subargument $\langle \mathcal{A}, \Gamma \rangle$ of $\langle \mathcal{A}_2, \Gamma_2 \rangle$ such that Γ and Γ_1 disagree.

It is possible to base a preference criterion on how strong the assumptions made in the derivations of the literals in the conclusion of an argument structure are. Let R be a temporal rule with temporal literal sequence $\mathcal{L}$ and with temporal equation $\mathcal{D} \cap \mathcal{B} \neq \emptyset$. Let $\langle \mathcal{L}, \mathcal{D} \rangle$ be a complete multi-configuration pair. The temporal constraint of R is $\mathcal{B}$ and its assumption is $\langle \mathcal{L}, \mathcal{D} \cap \mathcal{B} \rangle$. As discussed before, the number of groups of situations denoted by a multi-configuration pair is the same as the number of configurations in its configuration set. Let the cardinality of $\mathcal{D}$ be denoted by the natural number l_1 and let the cardinality of $\mathcal{D} \cap \mathcal{B}$ be denoted by the natural number l_2. This means that there are l_1 groups of possible situations in the “real world” on basis of the temporal literals in $\mathcal{L}$ and l_2 of them meet the constraints.

Consider now a case where $l_1 = l_2$. In that case, there are no configurations in $\mathcal{D}$, which are not in $\mathcal{B}$, i.e. all configurations in $\mathcal{D}$ meet the constraints. This means that all possible situations in the “real world” meet the constraints. This furthermore means that there is no chance that in the actual situation the constraints are not met. Therefore, the assumption made when deriving the head of the rule is as strong as possible. The assumption strength in this case would be $\frac{l_2}{l_1} = 1$, which is indeed the highest number for assumption strength possible. When the assumption strength of a rule is 1, this means that actually no assumption needs to be made since it is sure that constraints are met.

Now consider a case in which $l_1 > l_2$. In that case, there are configurations in $\mathcal{D}$, that are not in $\mathcal{B}$, i.e. there are configurations in $\mathcal{D}$ that do not meet the constraints. This means that there are possible situations in the “real world” in which constraints are not met. Therefore the assumption strength should be lower than in the case where all possible situations meet the constraints. The assumption strength in this case would indeed be lower than 1, since if $l_1 > l_2$, then $\frac{l_2}{l_1} < 1$.

The strength of the assumption made when the head of a rule is derived, should depend on the number of groups of possible situations in the "real world" relative to the number of groups of possible situations that meet the constraints and are thus in the assumptions. If in a rule the number of groups of possible situations is much higher than the number of groups of possible situations that meet the constraints, then in general the chance that the actual situation meets the constraints is low. Therefore the assumption made in the rule is weak and the rule has a low assumption strength. If in a rule the number of groups of possible situations is a bit or not at all higher than the number of groups of possible situations that meet the constraints, then in general the chance that the actual situation meets the constraints is high. Therefore the assumption made in the rule is strong and the rule has high assumption strength.

Since assumptions are expressed by multi-configuration pairs, the assumption strength of a rule is defined as the assumption strength of the multi-configuration pair that is its assumption. Assumption strength of argument structures is defined as well. A preference criterion is subsequently based on these definitions.

DEFINITION 4.37 (ASSUMPTION STRENGTH OF MULTI-CONFIGURATION PAIRS)

Let $\langle \mathcal{L}, \mathcal{D}_1 \rangle$ be a multi-configuration pair. Let $\langle \mathcal{L}, \mathcal{D}_2 \rangle$ be a complete multi-configuration pair. The rational n is said to be the assumption strength of $\langle \mathcal{L}, \mathcal{D}_1 \rangle$ if and only if $n = \frac{|\mathcal{D}_1|}{|\mathcal{D}_2|}$.

Let R be a temporal rule and let $\langle \mathcal{L}, \mathcal{D}_1 \rangle$ be its assumption. n is said to be the assumption strength of R.

REMARKS

Assumption strength is in no way implied to be an exact measurement of the chance that an assumption is "true", since in an interval set some intervals may have a higher chance of being the interval at which a predicate is actually true in the "real world" than others. This information is however not taken into consideration in the system and therefore assumption strength cannot be seen as anything more than a rough estimation of the chance that the actual situation is as in the assumption.

EXAMPLE 4.34 (ASSUMPTION STRENGTH)

The following are assumptions of rules and their assumption strength:

- $\left\langle \begin{array}{c}\langle Q(c,\{(1,2),(3,4)\})\rangle,\\ \times(\{(1,2),(3,4)\}) \cap \{\langle i\rangle \mid ends(i) < 5\}\end{array}\right\rangle$ and $\frac{2}{2} = 1$
- $\left\langle \begin{array}{c}\langle Q(c,\{(3,4),(5,7)\}), R(c,\{(2,6),(9,12)\})\rangle,\\ \{(3,4),(5,7)\} \times \{(2,6),(9,12)\} \cap \{\langle i_1,i_2\rangle \mid during(i_1,i_2)\}\end{array}\right\rangle$ and $\frac{1}{4}$
- $\left\langle \begin{array}{c}\langle Q(c,\{(1,2),(5,7)\}), R(c,\{(9,10),(20,39)\}), S(c,\{(3,5),(8,10)\})\rangle,\\ \{(1,2),(5,7)\} \times \{(9,10),(20,39)\} \times \{(3,5),(8,10)\} \cap\\ \left\{\langle i_1,i_2,i_3\rangle \,\middle|\, \begin{array}{c} before(i_1,i_2) \wedge \\ before(i_3,i_2) \wedge \\ before(i_1,i_3)\end{array}\right\}\end{array}\right\rangle$ and $\frac{4}{8}$

To derive literals, multiple assumptions may need to be made. The assumption strength of an argument structure for a set of literals should be based on all assumptions of the temporal rules used to derive them. The situations denoted by all assumptions of all temporal rules in an argument structure can be denoted by their combined multi-configuration pair. The assumption strength of an argument structure is thus the assumption strength of this combined multi-configuration pair.

DEFINITION 4.38 (ASSUMPTION STRENGTH OF ARGUMENT STRUCTURES)

Let $\langle \mathcal{A}, \Gamma\rangle$ be an argument structure. Let $\{R_1, \dots, R_n\}$ be the set of all temporal rules in $\mathcal{A}$. Let $\langle \mathcal{L}_1, \mathcal{D}_1\rangle, \dots, \langle \mathcal{L}_n, \mathcal{D}_n\rangle$ be the assumptions of $R_1, \dots, R_n$ respectively. Let $\langle \mathcal{L}, \mathcal{D}\rangle$ be the combined multi-configuration pair of $\langle \mathcal{L}_1, \mathcal{D}_1\rangle, \dots, \langle \mathcal{L}_n, \mathcal{D}_n\rangle$. Let n be the assumption strength of $\langle \mathcal{L}, \mathcal{D}\rangle$. n is said to be the assumption strength of $\langle \mathcal{A}, \Gamma\rangle$.

On basis of definition 4.38 a preference criterion is defined. Using this preference criterion, argument structures are preferred that are based on stronger assumptions. It can freely be adjusted and augmented to prefer for instance sets of argument structures based on more strict rules or more facts.

DEFINITION 4.39 (PREFERENCE CRITERION)

Let $\langle \mathcal{A}_1, \Gamma_1\rangle$ and $\langle \mathcal{A}_2, \Gamma_2\rangle$ be argument structures. $\langle \mathcal{A}_1, \Gamma_1\rangle$ is preferred over $\langle \mathcal{A}_2, \Gamma_2\rangle$, denoted $\langle \mathcal{A}_1, \Gamma_1\rangle \succ \langle \mathcal{A}_2, \Gamma_2\rangle$ if and only if the assumption strength of $\langle \mathcal{A}_1, \Gamma_1\rangle$ is greater than the assumption strength of $\langle \mathcal{A}_2, \Gamma_2\rangle$.

Proper and blocking defeat in the proposed logic is defined the same as in (García & Simari, 2004). The only difference is that argument structures have a set of literals as their conclusion.

DEFINITION 4.40 (PROPER DEFEATER)

Let $\langle\mathcal{A}_1, \Gamma_1\rangle$ and $\langle\mathcal{A}_2, \Gamma_2\rangle$ be two argument structures. $\langle\mathcal{A}_1, \Gamma_1\rangle$ is a proper defeater for $\langle\mathcal{A}_2, \Gamma_2\rangle$ at the literals in Γ, if and only if there exists a subargument $\langle\mathcal{A}, \Gamma\rangle$ of $\langle\mathcal{A}_2, \Gamma_2\rangle$ such that $\langle\mathcal{A}_1, \Gamma_1\rangle$ counterargues $\langle\mathcal{A}_2, \Gamma_2\rangle$ at the literals in Γ and $\langle\mathcal{A}_1, \Gamma_1\rangle \succ \langle\mathcal{A}, \Gamma\rangle$.

Corresponding to (García & Simari, 2004), an argument structure is a blocking defeater of another argument structure if it attacks it and the argument structures are unrelated by the preference order.

DEFINITION 4.41 (BLOCKING DEFEATER)

Let $\langle\mathcal{A}_1, \Gamma_1\rangle$ and $\langle\mathcal{A}_2, \Gamma_2\rangle$ be two argument structures. $\langle\mathcal{A}_1, \Gamma_1\rangle$ is a blocking defeater for $\langle\mathcal{A}_2, \Gamma_2\rangle$ at the literals in Γ, if and only if there exists a subargument $\langle\mathcal{A}, \Gamma\rangle$ of $\langle\mathcal{A}_2, \Gamma_2\rangle$ such that $\langle\mathcal{A}_1, \Gamma_1\rangle$ counterargues $\langle\mathcal{A}_2, \Gamma_2\rangle$ at the literals in Γ and $\langle\mathcal{A}_1, \Gamma_1\rangle$ is unrelated by the preference order to $\langle\mathcal{A}, \Gamma\rangle$, i.e. $\langle\mathcal{A}_1, \Gamma_1\rangle \nsucc \langle\mathcal{A}, \Gamma\rangle$, and $\langle\mathcal{A}, \Gamma\rangle \nsucc \langle\mathcal{A}_1, \Gamma_1\rangle$.

In accordance with (García & Simari, 2004), a set of argument structures defeats another set of argument structures if and only if it is its proper or blocking defeater.

DEFINITION 4.42 (DEFEATER)

Let $\langle\mathcal{A}_1, \Gamma_1\rangle$ and $\langle\mathcal{A}_2, \Gamma_2\rangle$ be argument structures. $\langle\mathcal{A}_1, \Gamma_1\rangle$ is a defeater for $\langle\mathcal{A}_2, \Gamma_2\rangle$, if and only if either:

- $\langle\mathcal{A}_1, \Gamma_1\rangle$ is a proper defeater for $\langle\mathcal{A}_2, \Gamma_2\rangle$; or
- $\langle\mathcal{A}_1, \Gamma_1\rangle$ is a blocking defeater for $\langle\mathcal{A}_2, \Gamma_2\rangle$.

In the next section, the dialectics of the proposed logic are defined and discussed.

4.5 DIALECTICS

The argumentation process in the proposed logic is very similar to that of (García & Simari, 2004). There are no major differences between the definitions in the proposed logic and the definitions in (García & Simari, 2004). Most differences that do exist stem from the fact that argument structures have a set of literals as their conclusion and do not interfere with the way the dialectics in DeLP work. A difference of another nature occurs in the definition of concordance. This definition is the only one in this subsection that has been adapted to a greater extent. While the similarities between the argumentation process in the proposed logic and the argumentation process in DeLP are great, all definitions were still included to make the proposed logic complete.

The definition of an argumentation line is the same, except for the conclusions of arguments.

DEFINITION 4.43 (ARGUMENTATION LINES)

Let $\mathcal{P}$ be a de.l.p. and $\langle\mathcal{A}_0, \Gamma_0\rangle$ an argument structure obtained from $\mathcal{P}$. An argumentation line for $\langle\mathcal{A}_0, \Gamma_0\rangle$ is a sequence of argument structures from $\mathcal{P}$, denoted $\Lambda = [\langle\mathcal{A}_0, \Gamma_0\rangle, \langle\mathcal{A}_1, \Gamma_1\rangle, \langle\mathcal{A}_2, \Gamma_2\rangle, \dots]$, where each element of the sequence $\langle\mathcal{A}_i, \Gamma_i\rangle$, $i > 0$, is a defeater of its predecessor $\langle\mathcal{A}_{i-1}, \Gamma_{i-1}\rangle$.

The notions of supporting and interfering argument structures have not been changed in the proposed logic.

DEFINITION 4.44 (SUPPORTING AND INTERFERING ARGUMENT STRUCTURES)

Let $\Lambda = [\langle\mathcal{A}_0, \Gamma_0\rangle, \langle\mathcal{A}_1, \Gamma_1\rangle, \langle\mathcal{A}_2, \Gamma_2\rangle, \dots]$ an argumentation line, we define the set of supporting argument structures $\Lambda_S = \{\langle\mathcal{A}_0, \Gamma_0\rangle, \langle\mathcal{A}_2, \Gamma_2\rangle, \langle\mathcal{A}_4, \Gamma_4\rangle, \dots\}$, and the set of interfering argument structures $\Lambda_I = \{\langle\mathcal{A}_1, \Gamma_1\rangle, \langle\mathcal{A}_3, \Gamma_3\rangle, \dots\}$.

The notion of concordance was introduced by (García & Simari, 2004) to avoid circular argumentations in which arguments are reintroduced in an argumentation line, making it infinite. In (García & Simari, 2004) argument structures are concordant if their sets of rules together with the set of strict rules from a de.l.p. is not contradictory (see page 31). As mentioned before, using only strict rules, arguments may be constructed for literals that should be defeasible. Therefore, in the definition of concordance the set of literals that have a strict derivation are used instead.

DEFINITION 4.45 (CONCORDANCE)

Let $\mathcal{P} = (\Pi, \Delta)$ be a de.l.p.. Let Σ be the set of literals that have a strict derivation from $\mathcal{P}$. Two arguments $\langle\mathcal{A}_1, \Gamma_1\rangle$ and $\langle\mathcal{A}_2, \Gamma_2\rangle$ are concordant iff the set $\Sigma \cup \mathcal{A}_1 \cup \mathcal{A}_2$ is non-contradictory and their assumptions are compatible. More generally, a set of argument structures $\{\langle\mathcal{A}_i, \Gamma_i\rangle\}_{i=1}^{n}$ is concordant iff $\Sigma \cup \bigcup_{i=1}^{n} \mathcal{A}_i$ is non-contradictory.

The notion of an acceptable argumentation line is again similar to (García & Simari, 2004).

DEFINITION 4.46 (ACCEPTABLE ARGUMENTATION LINE)

Let $\Lambda = [\langle \mathcal{A}_1, \Gamma_1 \rangle, \dots, \langle \mathcal{A}_i, \Gamma_i \rangle, \dots, \langle \mathcal{A}_n, \Gamma_n \rangle]$ be an argumentation line. Λ is an acceptable argumentation line iff:

1. Λ is a finite sequence
2. The set Λ_S, of supporting arguments is concordant, and the set Λ_I of interfering arguments is concordant.
3. No argument $\langle \mathcal{A}_k, \Gamma_k \rangle$ in Λ is a subargument of an argument $\langle \mathcal{A}_i, \Gamma_i \rangle$ appearing earlier in Λ $(i < k)$.
4. For all i, such that the argument $\langle \mathcal{A}_i, \Gamma_i \rangle$ is a blocking defeater for $\langle \mathcal{A}_{i-1}, \Gamma_{i-1} \rangle$, if $\langle \mathcal{A}_{i+1}, \Gamma_{i+1} \rangle$ exists, then $\langle \mathcal{A}_{i+1}, \Gamma_{i+1} \rangle$ is a proper defeater of $\langle \mathcal{A}_i, \Gamma_i \rangle$.

Dialectical trees are again defined similar to (García & Simari, 2004).

DEFINITION 4.47 (DIALECTICAL TREE)

Let $\langle \mathcal{A}_0, \Gamma_0 \rangle$ be an argument structure from a program $\mathcal{P}$. A dialectical tree for $\langle \mathcal{A}_0, \Gamma_0 \rangle$, denoted $\mathcal{T}_{\langle \mathcal{A}_0, \Gamma_0 \rangle}$, is defined as follows:

- The root of the tree is labeled with $\langle \mathcal{A}_0, \Gamma_0 \rangle$.
- Let N be a non-root node of the tree labeled $\langle \mathcal{A}_n, \Gamma_n \rangle$, and $\Lambda = [\langle \mathcal{A}_0, \Gamma_0 \rangle, \langle \mathcal{A}_1, \Gamma_1 \rangle, \langle \mathcal{A}_2, \Gamma_2 \rangle, \dots, \langle \mathcal{A}_n, \Gamma_n \rangle]$ the sequence of labels of the path from the root to N. Let $\langle \mathcal{B}_1, \mathrm{H}_1 \rangle, \langle \mathcal{B}_2, \mathrm{H}_2 \rangle, \dots, \langle \mathcal{B}_k, \mathrm{H}_k \rangle$ be all the defeaters for $\langle \mathcal{A}_n, \Gamma_n \rangle$.
  For each defeater $\langle B_i, \mathrm{H}_i \rangle$ $(1 \leq i \leq k)$, such that, the argumentation line
  $\Lambda' = [\langle \mathcal{A}_0, \Gamma_0 \rangle, \langle \mathcal{A}_1, \Gamma_1 \rangle, \langle \mathcal{A}_2, \Gamma_2 \rangle, \dots, \langle \mathcal{A}_n, \Gamma_n \rangle, \langle B_i, \mathrm{H}_i \rangle]$ is acceptable, then the node N has a child N_i labeled $\langle B_i, \mathrm{H}_i \rangle$.
  If there is no defeater for $\langle \mathcal{A}_n, \Gamma_n \rangle$ or there is no $\langle B_i, \mathrm{H}_i \rangle$ such that Λ' is acceptable, then N is a leaf.

The nodes in a dialectical tree are marked exactly as described in (García & Simari, 2004).

PROCEDURE 4.1 (MARKING OF A DIALECTICAL TREE)

Let $\mathcal{T}_{\langle \mathcal{A}, \Gamma \rangle}$ be a dialectical tree for $\langle \mathcal{A}, \Gamma \rangle$. The corresponding marked dialectical tree denoted $\mathcal{T}^*_{\langle \mathcal{A}, \Gamma \rangle}$, will be obtained marking every node in $\mathcal{T}_{\langle \mathcal{A}, \Gamma \rangle}$ as follows:

- All leaves in $\mathcal{T}_{\langle \mathcal{A}, \Gamma \rangle}$ are marked as "U"'s in $\mathcal{T}^*_{\langle \mathcal{A}, \Gamma \rangle}$.

- Let $\langle \mathcal{B}, \mathrm{H} \rangle$ be an inner node of $\mathcal{T}_{\langle \mathcal{A}, \Gamma \rangle}$. Then $\langle \mathcal{B}, \mathrm{H} \rangle$ will be marked as "U" in $\mathcal{T}^*_{\langle \mathcal{A}, \Gamma \rangle}$ iff every child of $\langle \mathcal{B}, \mathrm{H} \rangle$ is marked as "D". The node $\langle \mathcal{B}, \mathrm{H} \rangle$ will be marked as "D" in $\mathcal{T}^*_{\langle \mathcal{A}, \Gamma \rangle}$ iff it has at least a child marked as "U".

At the end of the dialectical process, it is of course necessary to obtain an answer on a query for a literal. In (García & Simari, 2004) a literal is warranted if the root of its associated marked dialectical tree is marked as "U". This stays the same in the proposed logic. For a user it may in addition be important to know on what assumptions the warrant of a literal is based. If literals are warranted, then this does not only include the assumptions on which the derivations of the literals are based, but also the assumptions on which their supporting arguments are based. If literals are not warranted, then this is based on the assumptions of the interfering arguments in the path in the tree in which the argument structure with the literals in its conclusion is defeated.

DEFINITION 4.48 (ASSUMPTIONS OF NODES IN A DIALECTICAL TREE)

Let $\langle \mathcal{A}, \Gamma \rangle$ be an argument structure and let $\mathcal{T}^*_{\langle \mathcal{A}, \Gamma \rangle}$ be its associated marked dialectical tree. Let $\langle \mathcal{A}_i, \Gamma_i \rangle$ be a node in $\mathcal{T}^*_{\langle \mathcal{A}, \Gamma \rangle}$. The supporting or interfering assumptions of $\langle \mathcal{A}_i, \Gamma_i \rangle$ are the following:

- If $\langle \mathcal{A}_i, \Gamma_i \rangle$ is a leaf, then its supporting assumption is the assumption of $\langle \mathcal{A}_i, \Gamma_i \rangle$ itself, and
- If $\langle \mathcal{A}_i, \Gamma_i \rangle$ is not a leaf and it is marked "D", then its interfering assumption is a disjunction of the supporting arguments of its children that are marked "U".
- If $\langle \mathcal{A}_i, \Gamma_i \rangle$ is not a leaf and it is marked "U", then its supporting assumption is a conjunction of the assumption of $\langle \mathcal{A}_i, \Gamma_i \rangle$ itself and the interfering assumptions of each of its children

Argument structures in the proposed logic can have multiple literals in their conclusion. This provides an extra advantage, namely that a user can query multiple literals. These are warranted together if they together are in the conclusion of an argument structure that emerges undefeated from the argumentation process. One query for multiple literals will only be warranted if they can be derived based on compatible assumptions. This is a consequence of the fact that literals derived using incompatible assumptions cannot be in the same argument according to definition 4.33. If one query is used for multiple literals and these literals together are not warranted, then it may be the case that they each (or some of them) are warranted if multiple queries are used, since in that case their assumptions do not necessarily need to be compatible.

DEFINITION 4.49 (STRICTLY WARRANTED LITERALS)

Let $\langle \mathcal{A}, \Gamma \rangle$ be an argument structure and $\mathcal{T}^*_{\langle \mathcal{A}, \Gamma \rangle}$ its associated marked dialectical tree. The literals in Γ are warranted under its supporting assumptions if and only if the root of $\mathcal{T}^*_{\langle \mathcal{A}, \Gamma \rangle}$ is marked as "U".

In (García & Simari, 2004) a query for a literal h is answered YES if h is warranted and NO if its complement is warranted. As discussed before, a complement of a temporal literal does not necessarily contradict it. The definition of an answer to a query is adapted accordingly.

DEFINITION 4.50 (ANSWER TO QUERIES) (GARCÍA & SIMARI, 2004)

There are four possible answers for a query of a set of literals Γ:

- YES under assumption α, if Γ is warranted under assumption α
- NO under assumption α if a set of literals H is warranted under assumption α and $\mathrm{H} \cap \Gamma$ is contradictory
- $UNDECIDED$, neither Γ, not a set of literals H, such that $\mathrm{H} \cap \Gamma$ is contradictory, is warranted
- $UNKNOWN$, if Γ contains one or more literals that are not in the language of the program

5 PROOF OF CONCEPT IMPLEMENTATION

To demonstrate the feasibility of the proposed temporal argumentation logic, a proof of concept implementation was made using SWI-Prolog. Prolog was chosen as the language of the implementation since it is a high-level language in which it is possible to describe complex and abstract things and thus is very suitable for rapid prototyping (Blackburn, Bos, & Kristina, 2006). The properties stated above make it possible to translate the definitions in this thesis reasonably direct into the language of the implementation, which is of course an advantage in a proof of concept implementation.

The sole purpose of this implementation of the proposed logic is to demonstrate the feasibility of the proposed logic in a medical diagnostic system. This means that no effort was made to make the implementation efficient or fast. In addition, the program is not complete. The argumentation process of the proposed logic was not included in the implementation, since it is based on and very closely resembles the argumentation process of DeLP. García and Simari already made a fine implementation of DeLP and thus demonstrated the feasibility of the argumentation process. The proposed logic is implemented up until the point it can be determined which of two counterarguments is stronger according to the preference criterion.

Functions on interval sets and intervals are defined in the proposed logic as being equal to their outputs, e.g. $before(\{(2,3)\}) = \{(0,1)\}$. It is however not possible to define a function in this way in Prolog. In Prolog, the implementation of the function $before$ on an interval set should have two arguments, one being its input and one being its output. The same problems appear when functions are used to define a temporal criterion in a temporal rule. These problems are clearly caused by the choice of Prolog as the language of the implementation and not by the proposed logic itself. They may also indicate that Prolog should not be the language of choice for an implementation of the proposed logic in a medical diagnosis system. The purpose of implementing the proposed logic is to demonstrate its feasibility and not to solve the issues Prolog has with functions. Since the functions on interval sets denote interval sets, all temporal arguments of temporal literals are interval sets and they are not functions on interval sets. The same is the case for temporal constraints; they are configuration sets in the implementation and are not defined by functions on intervals.

In the implementation for each of the definitions in this thesis, a Prolog-predicate was made. In some cases additional predicates had to be written to support the predicates implementing the definitions. It was tried to translate the definitions as literally as possible into Prolog, which succeeded in most cases. In the example below, such a literal translation of definition 4.8 into a Prolog-predicate is shown.

EXAMPLE 5.1 THE PROLOG-PREDICATE FOR STRICT RULES

The prolog-code below is used to implement definition 4.8.

```
% strict_rule(+Rule) is true if Rule meets the
% requirements of a strict rule as defined in
% definition 4.8.

strict_rule(Literal1<-[Literal2]):-
     literal(Literal1),
     literal(Literal2).

strict_rule(Literal1<-[Literal2|T]):-
     literal(Literal2),
     strict_rule(Literal1<-T).
```

In the proof of concept implementation, it is possible to include rules containing variables as long as these variables are not part of the temporal equation of a rule. This is due to the aforementioned problems with Prolog and the way functions are defined in the proposed logic. There is no way to make a function equal to its output and thus there is no way to define temporal constraints by using functions on intervals or to define a function determining the configuration set of the complete configuration pair of the temporal literals in a rule. There are probably various ways to circumvent this problem, it was however decided not to use them, since this would again just be the solving of the problems Prolog has with functions and this would not contribute to demonstrating the feasibility of the proposed logic.

Each definition implemented in the proof of concept implementation was tested using their corresponding examples from the section above. In cases there were not enough examples available to test an implemented definition, additional examples were made. The tests of each of the implemented definitions resulted in the desired and expected outputs. The proof of concept implementation thus demonstrated the feasibility of the proposed logic.

6 THE PROPOSED TEMPORAL ARGUMENTATION LOGIC AND THE REQUIREMENTS

In section 2 some requirements for a medical diagnosis program were established. In this section, the fulfilling of each of these requirements by a system that contains the proposed logic is discussed. Each section in this chapter treats one of the corresponding requirements described in chapter 2.

6.1 INCOMPLETE INFORMATION

As discussed in chapter 2.1, incomplete information is one of the most important things a medical diagnostic program should be able to handle. The proposed logic is an argumentation logic. In argumentation logic, when information is incomplete literals are derived on basis of the information that is available. If some of the missing information becomes available later, the derived literals may be defeated. Missing information is essentially handled as in the Tweety-example at page 12. If information is missing that Tweety is a penguin, then it will be derived that Tweety flies, since this is generally the case for birds. If the missing information that Tweety is a penguin becomes available, then "Tweety flies" may be defeated.

Consider a defeasible rule stating that "if someone sneezes, then they generally have a cold", a defeasible rule stating "if someone has a high fever then they generally do not have a cold" and a defeasible rule stating that "if someone sneezes and has a fever, then they generally have the flu". Suppose that Mary sneezes and that information on whether she has a fever is missing. In that case, it may be derived that she has a cold, which is the most probable in case nothing is known about whether Mary has a fever[12]. If information that she has a high fever becomes available, then "Mary does not have a cold" may be derived, defeating "Mary has a cold". Furthermore, it may be derived that Mary has the flu. If it is not feasible to have rules such as the second rule in the program, both "Mary has a cold" and "Mary has the flu" may be derived. Then a choice has to be made between whether Mary has only a cold, only the flu, both or none of both. This is the problem of multiple or single diseases stated in section 2.3, a way of choosing one of the alternatives is proposed in section 6.3.

Another way in which incomplete information can be handled is by using negation as failure more explicit. When negation as failure is used, it is assumed that a literal is not true when it cannot be derived that it is. Again, when more information becomes available from which it can be derived that the literal is true, this conflict is taken care of by the argumentation process. García and Simari propose an extension of DeLP in which negation as failure may be used. There are no obvious obstructions to extending the proposed logic with negation as

[12] If it is not the case that it is the most probable that someone has a cold if they sneeze and no information about whether they have a fever is available, then the rule should be adapted to state "If someone sneezes and does not have a fever, then they have a cold".

failure in the same way. Further research should however establish whether this is in fact possible.

It may be possible to extend the proposed logic to deal with incomplete information even better. To do this, all unknown facts may get a default value assigned; the value they are most likely to have if they are unknown. Subsequently a preference criterion may be defined in which the defaults and literals derived on basis of these defaults may be defeated when contradictions arise with literals that are known facts or derived on basis of (more) known facts. García and Simari propose an extension of DeLP with presumptions. These presumptions may be used as defaults. Again, there are no obvious obstructions to extending the proposed logic in the same way, but further research should establish whether this is really the case.

A system implementing the proposed logic can in addition be extended with a feature that computes the informative value of missing information and that subsequently requests the most informative missing information from the user. Though such a feature may be very useful, logic probably will not be the most suitable way to realize it. The proposed logic may however be used by the program to gain information about how informative certain missing information is. Whether and what changes need to be made to the proposed logic to serve this purpose may be investigated in future research.

The way in which completely or partially missing temporal information is handled in the proposed logic is discussed in section 6.5.

6.2 CONTRADICTIONS

Argumentation logic in general is very well suited to deal with contradictions. If a contradiction is derived in an argumentation logic one of the contradictory literals is defeated by the other, depending on the preference criterion. How well a system incorporating the proposed logic handles contradictions does thus depend on the choice of the preference criterion.

The proposed logic does not distinguish contradictions that arise due to mistakes. It is however possible to make the argumentation process clear to users. If necessary, they can identify contradictions due to mistakes and correct them by adding additional facts and rules or change them. When a contradiction arises because of disagreement in the medical domain, then the preference criterion may be adapted to suit the preferences of the user of the program.

6.3 SINGLE OR MULTIPLE DISEASES

In the proposed logic, it is possible that argument structures have multiple conclusions. This means that a user may make a query for multiple literals or a single one. A user may determine on basis of the order the argument structures for multiple or single literals have according to the preference criterion which of the argument structures is “stronger”. This may aid them in determining whether it is more probable that a patient has multiple diseases or suffers just from a single one. The preference criterion may for instance be based on the strength of assumptions, specificity or the strength of the rules used in the derivation of the literals. In addition, the preference criterion may be adapted to prefer argument structures having certain combinations or numbers of literals in its conclusion. Additional facts and rules may be added to the system if more certainty is needed.

In the example in section 2.3 three possibilities were considered. The first was that the patient has only $disease_1$, the second was that the patient only has $disease_2$ and the third was that the patient has both diseases. In the proposed logic for each of the possibilities an argument structure can be constructed, let them be $\langle \mathcal{A}_1, \{disease_1\}\rangle$, $\langle \mathcal{A}_2, \{disease_2\}\rangle$ and $\langle \mathcal{A}_3, \{disease_1, disease_2\}\rangle$ respectively. The third argument structure of course can only be constructed if the assumptions on which $disease_1$ and $disease_2$ were derived are compatible, otherwise, this could indicate that the patient probably does not have both diseases (see section 4.3). The argument structures above can be ordered according to some preference criterion. As mentioned above, this preference criterion could be based on several relevant factors and the choice of preference criterion clearly is vital in this case. Users could then use the order on the arguments above to determine which argument structure is “stronger” (i.e. is preferred according to the preference criterion). This in turn could help them determining whether it is more probable that the patient has $disease_1$, $disease_2$ or both diseases. It is important to remark here that the preference criterion does not give an absolute probability of whether a patient has multiple diseases or a single one, when it is chosen right it just gives a rough indication. In addition, the user of course ultimately has the responsibility of choosing the right diagnosis and the order on argument structures can just be supporting in making that choice.

It may be possible to automate the process of choosing between multiple or single diagnoses by treating their argument structures as if they attack each other. How this exactly should be realized may be subject to further research.

6.4 EXCLUSIONARY CRITERIA

In the proposed logic, it is possible to use exclusionary criteria to rule out certain conditions under certain circumstances. Exclusionary criteria can be formulated in the form of strict or defeasible rules in which the antecedent contains the exclusionary criterion and the consequent contains the negation of the literal that should not be derived in case the exclusionary criterion is true. Using such rules, if a literal that should be excluded is derived, its negation may also be derived, resulting in two counterarguments. In the argumentation process, the one of the two counterarguments may be defeated. In case the preference criterion is chosen correctly, this should be the argument structure for the literal that should be excluded.

Consider the example in section 2.4 once again. There may be two rules in a system implementing the proposed logic. The rule by which cholecystitis may be may be derived, viz. $cholecystitis \prec symptom_1, \dots, symptom_n$ and the rule by which cholecystitis may be excluded, viz. $\neg cholecystitis \leftarrow cholecystectomy$[13]. In the case the antecedents of both rules are true, two argument structures can be constructed in the proposed logic, namely $\langle \mathcal{A}_1, \{cholecystitis\}\rangle$ and $\langle \mathcal{A}_2, \{\neg cholecystitis\}\rangle$. Clearly these argument structures are counterarguments of each other, therefore $cholecystitis$ or $\neg cholecystitis$ is defeated. If the preference criterion is chosen correctly, $\neg cholecystitis$ will be preferred over $cholecystitis$ and $cholecystitis$ will be defeated.

As mentioned in section 2.4, exclusionary criteria may also be included in the antecedent of a rule. This is also not a problem since negations in the antecedents of rules are allowed.

6.5 TEMPORAL INFORMATION

It is possible in the proposed logic to formulate criterions on the temporal aspects of the literals in the antecedent of a rule. The relationships between intervals as described by (Allen, 1983) may be used for this purpose. These relationships describe all possible relationships between intervals, however if for some reason it is needed to describe additional criterions on temporal aspects of literals, there are no obvious obstacles to adding the appropriate functions to the language of the logic.

The proposed logic distinguishes itself from other prior existing argumentation logics by incorporating a way to express partial temporal information and by providing the possibility to reason with this partial information.[14] Partial temporal information can be expressed in complex ways if necessary and may depend on partial or complete temporal information about other literals. In addition to the functions on interval sets already defined for this purpose, there are no obvious obstructions to defining supplementary functions, such as $thedaybefore(\mathcal{I})$ or $hours\_after(8, \mathcal{I})$, where $\mathcal{I}$ denotes an interval set.

The dependency of the description of partial temporal information about a literal on the temporal information of another literal may however in some cases cause difficulties. Consider the following facts: $fever(mary, \mathcal{I})$ and $headache(mary, \mathcal{K})$, where $\mathcal{I} = before(\mathcal{K})$ and $\mathcal{K} = after(\mathcal{I})$. For these facts, it may not be immediately possible in the proposed logic to determine which intervals are exactly in their temporal literal sequences. There are several solutions to such a problem. The first is to consider it as a mistake in the input of a system incorporating the proposed logic and not as a problem of the logic itself. The second solution could be to redefine the functions on literal sets in such a way that if the above occurs in the set of facts of a de.l.p., the result of the functions is a temporal literal

[13] For the sake of clarity, the rules are kept as simple as possible and no temporal terms are used.

[14] To my knowledge, at this time no other argumentation logic exists that can do the same. It may however be the case that such an argumentation logic did exist prior to my thesis and that I missed it during my research. In that case, this claim should be considered defeated.

sequence containing all intervals in the time span considered by the system [15]. The best way to detect cycles such as the one described above and the best solutions to them should be determined by future research.

A direct consequence of reasoning with partial temporal information is that assumptions may need to be made in the derivation of a literal. In the proposed logic “real” assumptions are only made if necessary and the exact assumptions on which the derivation of a literal is based can be made visible to a user (see section 6.7). A user can thus verify whether certain assumptions are reasonable from a medical point of view and in case they are not, or in cases additional temporal information has become available, users can adapt incorrect rules or add and change facts. One obvious disadvantage of making assumptions in the derivation of a literal is that there are greater chances that things are derived that are not true. It is however possible give an indication of the “strength” of an assumption to the user by using the notion of assumption strength of an argument structure (definition 4.38). Furthermore, uncertainty is inherent to reasoning with partial information, it is better to make this explicit than implicit. Uncertain information in most cases seems to be better than no information at all.

As discussed in the introduction of section 4, temporal information may not only be partially available or completely missing, it may also be the case that it is imprecise. In the proposed logic, imprecise temporal information may be reckoned with by making the temporal literal sequences of literals larger, i.e. by adding intervals. This may be done automatically by a part of a system incorporating the logic or by the user. Future research, (possibly in the domain of psychology) may focus on determining in which cases patients are imprecise in reporting temporal information and what intervals need to be added to an interval set in that case.

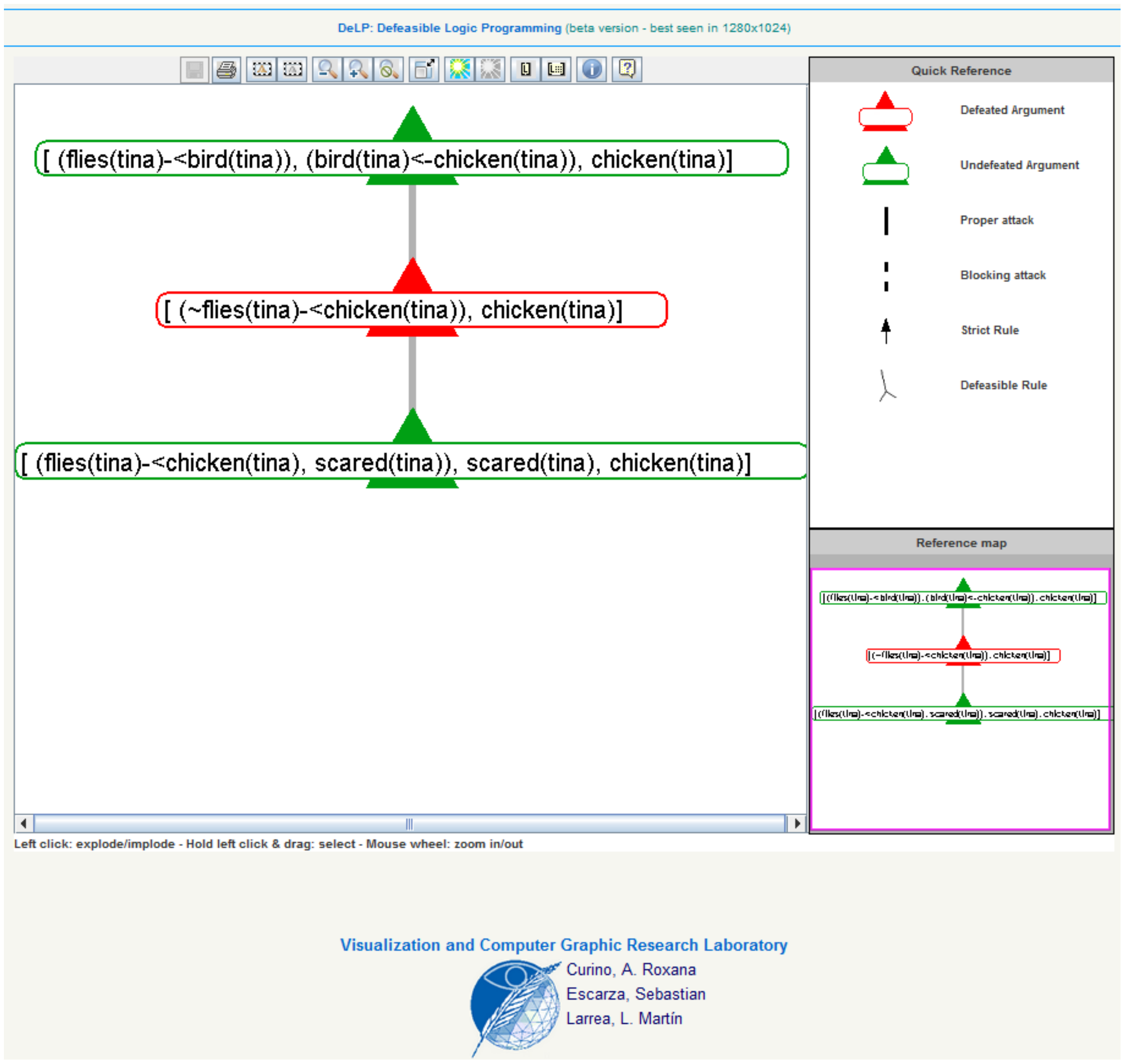
6.6 CHAINING RULES

In the proposed logic, clearly rules can be chained to derive a literal. In addition, there is no obstruction to using literals denoting diseases in the antecedent of a rule, since literals denoting symptoms and literals denoting diseases or other information are not distinguished from each other.

6.7 CLARIFYING THE REASONING PROCESS

The argumentation process of argumentation logic in general is very intuitive to users. In most cases, it is possible to clarify the argumentation process to users in some graphical way. The argumentation process in the proposed logic very closely resembles the argumentation process of DeLP. In the implementation of DeLP by García and Simari, the argumentation process is clarified to users graphically. Below is an example of the dialectical tree from their implementation for the query $flies(tina)$ from example 3.1.

[15] Except for for instance the last interval for the function $before$ and the first interval for the function $after$.


DeLP: Defeasible Logic Programming (beta version - best seen in 1280x1024)

Quick Reference

Defeated Argument

Undefeated Argument

Proper attack

Blocking attack

Strict Rule

Defeasible Rule

[(flies(tina)-<bird(tina)), (bird(tina)<-chicken(tina)), chicken(tina)]

[(~flies(tina)-<chicken(tina)), chicken(tina)]

[(flies(tina)-<chicken(tina), scared(tina)), scared(tina), chicken(tina)]

Reference map

Left click: explode/implode - Hold left click & drag: select - Mouse wheel: zoom in/out

Visualization and Computer Graphic Research Laboratory
Curino, A. Roxana
Escarza, Sebastian
Larrea, L. Martín

FIGURE 7 GRAPHICAL FRONT-END OF THE IMPLEMENTATION OF DELP BY GARCÍA AND SIMARI (AVAILABLE AT HTTP://LIDIA.CS.UNS.EDU.AR/DELP_CLIENT)

Different options for viewing the dialectical tree are available in the graphical front-end of the implementation of García and Simari, such as showing the complete argument of a structure or hiding it. This makes it even easier for users to understand the reasons of why certain literals are defeated or not. Since the dialectical process of the proposed logic is based upon the dialectical process as defined in (García & Simari, 2004) and does not deviate much from it, it should be possible to make the dialectical process of the proposed logic clear in a similar way.

In the proposed logic, the making of assumptions in a derivation and their contents are tracked very precisely and therefore it can be determined exactly on which assumptions the derivation of a literal is based. The syntax of a multi-configuration pair with its numerous brackets is suitable for a computer to reason with, it is however very unattractive to look at for a human. This problem could be solved by supplementing an implementation of the proposed logic with a component that presents the assumptions in a graphical way that is easier to understand by humans. Designing such a component should not be very hard since all assumptions have the exact same form. Which graphical representation of assumptions is the most appropriate can be subject to further research.

7 CONCLUSION AND SUGGESTIONS FOR FURTHER RESEARCH

7.1 EVALUATION OF THE PROPOSED ARGUMENTATION LOGIC

As discussed in section 6, the proposed logic meets all requirements stated in section 2 to a greater or lesser extent. There are, however, some additional advantages of using the proposed logic in a medical diagnosis system. One of these advantages is that it is possible to use it not only as a temporal argumentation logic, but also as just a temporal logic, just an argumentation logic or just as a logic. The first can be accomplished by using only strict rules and interval sets containing just one interval. The logic can be used in the second way by not including temporal rules and temporal literals. The last option can be accomplished by doing both, namely not including temporal rules and literals and using only strict rules. These different options make it possible for an implementation of the proposed logic to serve different purposes in a medical diagnostic system that is modular, such as a multi-agent system that incorporates multiple reasoning agents.

The most apparent disadvantage of the proposed logic is that an implementation probably will not be very efficient. In the proof of concept implementation described in section 5, no attention was paid to efficiency. It became however apparent quickly that the implementation is not very efficient. In cases, there is not much temporal information available and the time span considered by the system is great, interval sets may become very large, computing the output of functions on them and their intersections may then require a lot of computations. This is also the case for the computations of other things, such as the complete multi-configuration pair of a temporal literal sequence. In addition, the obtaining of the set of ground rules instead of using schematic rules is not efficient.

The size of an interval set does directly depend on the amount of temporal information that is available. The more information that is available, the smaller the interval sets are. In that light, needing to do more computations may just be considered a price to pay for reasoning with partial information. There may however be some other ways to reduce the number of computations. Heuristics may for instance be used to compute the output of functions on interval sets, to make interval sets smaller or to reduce the number of time points in the time span considered by the system. Since the emphasis of this thesis has not been on the efficiency of algorithms there may exist many more ways to reduce it. In addition, ways may be found to use schematic rules instead of ground ones. In the proof of concept implementation, schematic rules could be used when functions were not used in rules or facts. Since there is a problem with implementing functions in their usual way in Prolog, it may be the case that schematic rules containing functions may be used when another language is used to implement the proposed logic.

Overall, the proposed logic seems suitable to use in a medical diagnostic system. It meets many important requirements that should be met by a logic in a medical diagnostic system. The fact that partial temporal information can be expressed and reasoned with in the proposed logic is considered its most important and distinctive feature. A medical diagnostic system implementing the proposed logic is feasible as demonstrated by a proof of concept implementation. The computational complexity of an implementation may get high quickly when less temporal information is available. Ways to lower this computational complexity have not been investigated in this thesis and could be subject to further research.

7.2 SUGGESTIONS FOR FURTHER RESEARCH

In addition to the suggestions for further research given in sections 6 and 7.1, some other important and interesting matters could be investigated. The most important subject of further research that already has been mentioned is the computational complexity of an implementation, since this may be the weakest point of the proposed logic.

An additional interesting subject to investigate would be the suitability of the proposed logic in other domains than the medical domain. As described in section 6, the preference criterion of the proposed logic is crucial to how well a system incorporating it performs. It may therefore additionally be important to determine what the best preference criterion is from a medical point of view and whether this is different for different medical domains.

In the proposed logic, only temporal information may be partial. It would be very interesting and probably useful to see if it is possible to extend the proposed logic such that other types of partial or imprecise information can be reasoned with as well. It may also be interesting to see whether these other types of partial information can be incorporated in the logic the same way as is done with partial temporal information. An example would be a case in which it is unclear whether the blood pressure of a patient is normal or high (due to for instance an imprecise measurement). There is no obvious reason why a fact $blood\_pressure(patient, \{high, normal\})$ representing this information could not be expressed in the proposed logic and reasoned with. From my point of view, this is the most interesting subjects to investigate further.

8 REFERENCES

Aldo, A. G. (2010). *Non-monotonic Logic*. (E. N. Zalta, Editor) Retrieved August 1, 2011, from Stanford Encyclopedia of Philosophy (Summer 2010 Edition): http://plato.stanford.edu/archives/sum2010/entries/logic-nonmonotonic/

Allen, J. F. (1983, November). Maintaining knowledge about temporal intervals. *Communications of the ACM, 26*(11), 832-843.

Berlin, A., Sorai, M., & Sim, I. (2006, December). A taxonomic description of computer-based clinical decision support systems. *Journal of Biomedical Informatics, 39*(6), 656-667.

Berner, E. S., & La Lande, T. J. (2007). Development and Evaluation of Clinical Decision Support Systems. In E. S. Berner, *Clinical Decision Support Systems Theory and Practice.* New York: Springer.

Blackburn, P., Bos, J., & Kristina, S. (2006). *Learn Prolog Now!* (7 ed.). College Publications.

Bochman, A. (2007). Nonmonotonic reasoning. In D. M. Gabbay, & J. Woods (Eds.), *Handbook of the History of Logic* (Vol. 8, pp. 557-632). Elsevier.

Chesñevar, C. I., Maguitman, A. G., & Loui, R. P. (2000). Logical Models of Argument. *ACM Computing Surveys, 32*, 337-383.

García, J. A., & Simari, R. G. (2004, January). Defeasible Logic Programming An argumentative Approach. *Theory and Practice of Logic Programming, 4*(2), 95-138.

Garg, A. X., Adhikari, N. K., McDonald, H., Rosas-Arenallo, P., Devereaux, P. J., Beyene, J., . . . Haynes, B. R. (2005). Effects of Computerized Clinical Decision Support Systems on Practitioner Performance and Patient Outcomes. *The Journal of the American Medical Association*, 1223-1238.

Gartner, J., Swift, T., Tien, A., Damásio, C. V., & Pereira, L. M. (2000). Psychiatric Diagnosis from the Viewpoint of Computational Logic. In *Proceedings of the First International Conference on Computational Logic* (pp. 1362-1376). London, UK: Springer-Verlag.

Patel, V. L., Shortliffe, E. H., Stefanelli, M., Szolovits, P., Berthold, M. R., Bellazzi, R., & Abu-Hanna, A. (2009, May 1). The coming of age of artificial intelligence in medicine. *Artificial Intelligence in Medicine, 46*(1), 5-17.

Prakken, H., & Vreeswijk, G. (1998, April 9). Logics for Defeasible Argumentation.

Reichgelt, H., & Vila, L. (2005). Temporal qualification in artificial intelligence. In M. Fisher, D. Gabbay, & L. Vila, *Handbook of Temporal Reasoning in Artificial Intelligence* (pp. 167-194). Elsevier.

Schlechta, K. (2007). Nonmonotonic logics: a preferential approach. In D. M. Gabbay, & J. Woods (Eds.), *Handbook of the History of Logic* (Vol. 8, pp. 451-516). Elsevier.

The Merck Manual. (2007 (1), December). *Choledocholithiasis and Cholangitis: Gallbladder and Bile Duct Disorders*. (E. A. Shaffer, Editor) Retrieved July 27, 2011, from The Merck Manual: http://www.merckmanuals.com/professional/sec03/ch031/ch031e.html

The Merck Manual. (2007 (2), February). *Treatment of Pain: Pain*. (R. K. Portenoy, Editor) Retrieved July 20, 2011, from The Merck Manual: http://www.merckmanuals.com/professional/sec17/ch219/ch219c.html

The Merck Manual. (2008 (1), March). *Chronic and Recurrent Abdominal Pain: Approach to the Patient with Upper GI complaints*. (N. J. Greenberger, Editor) Retrieved July 20, 2011, from The Merck Manual: http://www.merckmanuals.com/professional/sec02/ch007/ch007b.html

The Merck Manual. (2008 (2), May). *Idiopathic Interstitional Pneumonias: Interstitial Lung Diseases*. (T. E. King, Editor) Retrieved July 25, 2011, from The Merck Manual: http://www.merckmanuals.com/professional/sec05/ch059/ch059b.html

The Merck Manual. (2009, March). *Reye's Syndrome: Miscellaneous Disorders in Infants and Children*. (E. J. Palumbo, Editor) Retrieved July 20, 2011, from The Merck Manual: http://www.merckmanuals.com/professional/sec20/ch304/ch304f.html

The Merck Manual. (2010, March). *Clinical Decision Making Strategies: Clinical Decision Making*. (D. L. McGee, Editor) Retrieved July 27, 2011, from The Merk Manual: http://www.merckmanuals.com/professional/sec23/ch348/ch348c.html

Vreeswijk, G. A. (1997, February). Abstract argumentation systems. *Artificial Intelligence, 90*, 225-279.

Wright, A., & Sittig, D. F. (2008, October). A four-phase model of the evolution of clinical decision support architectures. *International Journal of Medical Informatics, 77*(10), 641-649.

9 APPENDIX: OVERVIEW OF LOGICS CONSIDERED FOR A MEDICAL DIAGNOSTIC SYSTEM

In the search for a logic that could be used as a basis for a medical diagnostic system, several types of logic have been evaluated. Most of these types of logic have not been investigated into great depth, since in many cases it already appeared very early in the process that the type of logic did not have the required properties for a medical diagnostic system. In most cases, several logics of the same type were studied. For each type of logic a summary of its syntax and semantics was made, furthermore, ideas were sketched on how the logic could be used in a medical diagnostic system. Additionally, the advantages and disadvantages of each logic for a medical diagnostic system were described.

It was decided that it is not a good idea to include the complete output of these investigations in this appendix, since this would make this appendix far too massive. In this appendix, a very short description and a short summary of the evaluation of each of the types logics that have been investigated is provided. Only the most important properties that make a logic suitable or unsuitable for use in a medical diagnostic system are given. Nonmonotonic logic and argumentation logic are already discussed elaborately in this thesis and are therefore not discussed here.

The information in the descriptions of the types of logics in this appendix was obtained from the online version of the *Stanford Encyclopedia of Philosophy*[16].

HYBRID LOGIC

SHORT DESCRIPTION

In hybrid logic, additional expressive power is added to modal logic. While in normal modal logic it is only possible to say something about the world you are in now (at a given moment in the reasoning process) and the worlds accessible from it, in hybrid logic it is possible to denote and reason about specific worlds that are not directly accessible from the world you are currently in.

ADVANTAGES AND DISADVANTAGES

Hybrid logic does not provide a direct way to express or reason with incomplete information conveniently. There are some ideas on how to change the logic, such that it is possible to do this. These ideas all are not very practical and require a lot of information about diseases and their probabilities. In addition, almost all ideas require the combination of hybrid logic with other types of logic.

There is not a straightforward way to deal with contradictions in hybrid logic, the fact that it is possible to denote specific worlds may however be used to find a way to do this. There is a version of hybrid logic, called hybrid tense logic in which it is possible to reason with

[16] See: http://plato.stanford.edu

temporal information. This may be an advantage of this logic. Hybrid logic and modal logic in general seems to be harder to understand for humans than for instance classic logic, it may thus be harder to make the reasoning process of a medical diagnostic system that is based on this logic clear to users.

DEONTIC LOGIC

SHORT DESCRIPTION

In deontic logic, concepts as permissibility, obligatory and optionality can be expressed and reasoned with.

ADVANTAGES AND DISADVANTAGES

Several deontic logics have been investigated and were evaluated. Most of these logics did not have properties that are beneficial for a medical diagnosis program that for instance classic logic does not have. The most interesting was standard possible world Kripke semantics for standard deontic logic. In this semantics, a world j is accessible from a world i if everything that is obligatory in i is obligatory in j. When every symptom and property of a patient that is known is considered obligatory, then it would be possible to use this semantics to determine which worlds (and thus diseases) are possible according to the known symptoms, providing a way to incorporate a way of reasoning with incomplete information.

It is also possible that for diseases, certain symptoms have to be present (obligatory), while others may be present (optional) or may be impossible to occur with a disease (impermissible). Deontic logic could be used to model this. This would allow for a very intuitive way to include for instance exclusionary criteria.

In the case a contradiction is derived and one of the contradictory sentences is obligated and the other is not, this could be used to determine which of the sentences should be rejected. It is however not possible to deal with all kinds of contradictions in this manner. Furthermore, this would mean that the logic should be modified severely to be able to reject the right sentences.

MANY-VALUED LOGIC

SHORT DESCRIPTION

In many-valued logic, sentences can have more or other truth-values than only $true$ and $false$. The truth-value of a compound sentence is determined by the truth-values of its component sentences.

ADVANTAGES/DISADVANTAGES

Sentences in many-valued logic may not only get assigned the value $true$ or $false$, but also the value $unknown$. This may provide a way to define a reasoning process in which incomplete information can be dealt with in a sensible manner.

Contradictions in this logic may arise less often if more than the three truth-values mentioned above are used. It is however not clear what the semantics of a logic with these multiple truth-values should be and how these relate to the “real world” in which patients either have a symptom or a disease or do not[17]. Furthermore, this would probably make the logic very counterintuitive. Using multiple truth-values would thus probably make it harder to explain the reasoning process to users.

RELEVANCE LOGIC

SHORT DESCRIPTION

In normal logic, it is possible to make very counterintuitive inferences. An example of this is that in normal logic, from a premises p, the consequent $q \to q$ may be derived. In relevance logic, it is attempted to avoid some of these counterintuitive paradoxes of implication. More specific, in relevance logic, inferences in which consequents have completely different topics as their antecedents are excluded.

ADVANTAGES AND DISADVANTAGES

The main advantage of relevance logic is that inferences in it are more intuitive. This may make it easier to explain the reasoning process to users. On the other hand, it may make it impossible to derive things that are true and useful but just counterintuitive to humans.

PARACONSISTENT LOGIC

SHORT DESCRIPTION

In a paraconsistent logic, even when contradictions are derived, the inference relation does not explode into triviality. In a paraconsistent logic several principles of classic logic are abandoned, making the logic propositionally weaker than classic logic.

ADVANTAGES AND DISADVANTAGES

It is of course not desirable that inferences in a medical diagnostic system explode into triviality. This logic does not seem to have any other advantages for medical diagnostic systems.

FUZZY LOGIC

SHORT DESCRIPTION

Sentences in fuzzy logic have a degree of truth, classically on a continuous scale between 0 and 1.

ADVANTAGES AND DISADVANTAGES

[17] Many valued logic is not directly suitable to use to reason about probabilities.

It is possible to use truth degrees to denote the certainty that a patient has a certain symptom or a certain disease. This may provide a way to reason not only with in complete information, but also with information that is only partially missing.

The reasoning process of fuzzy logic is not very easy to understand and it is a very controversial logic.

FREE LOGIC

SHORT DESCRIPTION

In free logic, terms may denote objects outside the domain of discourse of a logic and may denote non-existing objects.

ADVANTAGES AND DISADVANTAGES

Free logic does not have any clear advantages over for instance classic logic to use in a medical diagnostic system.

SECOND AND HIGHER ORDER LOGIC

SHORT DESCRIPTION

Second and higher order logics are extensions of first order logic. In first order logic, variables and quantifiers range over elements of the domain of discourse. In second and higher order logic, variables may range over sets of such elements and sets of such sets and so on.

ADVANTAGES AND DISADVANTAGES

Second and higher order logics are more expressive than first order logic. There however seems no need for this kind of larger expressivity in a logic for a medical diagnostic system. Furthermore, the semantics of second and higher order logic seems to be more complicated and this makes it harder to make the reasoning process of a program implementing second or higher order logic clear to users.

LINEAR LOGIC

SHORT DESCRIPTION

In linear logic, certain formulas are marked by modals. The usual structural rules of contraction and weakening are only applicable to formulas marked by certain of these modals. Linear logic is a refinement of classic and Intuitionistic logic in which the emphasis is on formulas as resources.

ADVANTAGES AND DISADVANTAGES

Linear logic has many well-described applications in the field of AI, especially in the domain of knowledge representation. From that perspective, this logic could be interesting. The logic does however not seem to meet any of the requirements defined for a medical diagnostic program specifically.

EPISTEMIC LOGIC

SHORT DESCRIPTION

Epistemic logic is a modal logic in which notions as knowledge and believe can be reasoned with. In epistemic logic, it is possible to model scenarios about individuals and groups of individuals that know or believe certain things.

ADVANTAGES AND DISADVANTAGES

Epistemic logic would be suitable to use to reason with uncertain information. It may for instance be believed that a patient has a certain symptom, while this is not known for certain. This way of dealing with incomplete information does however not seem to be the most appropriate in the medical domain. There is no way to determine how certain or uncertain information is and the logic is more about the individuals that believe or know information than about the information itself. Epistemic logic seems not to have any other advantages for a medical diagnostic system.

PROPOSITIONAL DYNAMIC LOGIC

SHORT DESCRIPTION

Propositional dynamic logic is a modal logic that is mainly used to study the properties of computer programs. Propositional dynamic logic is concerned with the executions of programs and whether these programs do or do not derive certain formulas when they are in a certain state.

ADVANTAGES AND DISADVANTAGES

This logic does not seem to help meeting any of the determined requirements for a medical diagnostic program. It could be used to model the working of the human body and help determining what disease a patient has from that point of view, since it is concerned with the execution of programs and the human body could be modeled analogous to that. This idea is however very far-fetched and still many of the requirements are not met.

PROVABILITY LOGIC

SHORT DESCRIPTION

Provability logic is a modal logic in which reasoning about mathematical theories and what can be expressed in them is conducted.

ADVANTAGES AND DISADVANTAGES

Provability logic seems more suitable to use in research on the fundamentals of mathematical theories than to conduct more practical forms of reasoning. This logic does not seem to have any advantages when used in a medical diagnostic program.

INFINITARY LOGIC

SHORT DESCRIPTION

In infinitary logic, formulas are identified as infinite sets. In this logic, connectives such as disjunction and conjunction can be of infinite length.

ADVANTAGES AND DISADVANTAGES

While the expressive power of infinitary logic exceeds the expressive power of first order logic, expressive power is not increased in a way that would benefit a medical diagnostic system. This logic does not have any advantages for use in a medical diagnostic system.

INDEPENDENCE FRIENDLY LOGIC

SHORT DESCRIPTION

Independence friendly logic is an extension of first order logic in which more dependencies and independencies between individuals that are quantified over can be expressed.

ADVANTAGES AND DISADVANTAGES

This logic seems only to be useful in the fields of linguistics and math, it does not seem to have any advantages when used in a medical diagnostic program.

QUANTUM LOGIC

SHORT DESCRIPTION

In quantum logic, the theories of quantum mechanics are taken into account.

ADVANTAGES AND DISADVANTAGES

Quantum logic is very complicated and controversial; this makes it very unsuitable for use in a medical diagnosis system.