Terap-IA, a Knowledge-Based System for Pneumonia Treatment*

Pilar Barrufet

Consorci Sanitari de Mataró Hospitals
Lepant, 13. 08301 Mataró, Spain
e-mail: pbarrufet@csm.scs.es

Josep Puyol-Gruart, Carles Sierra

Artificial Intelligence Research Institute (IIIA)
Spanish Scientific Research Council (CSIC)
Campus UAB. 08193 Bellaterra, Spain
e-mail: {puyol,sierra}@iiia.csic.es

Abstract

Terap–IA is a knowledge-based system for the treatment of community-acquired pneumonia in adults. In this paper we concentrate on its description and implementation. The implementation has been made using **Milord II**, a modular language for knowledge-based systems. We will stress the uncertainty modelling of the domain, with special emphasis on the **Milord II** capabilities for that purpose.

1 Description of the problem

A community-acquired pneumonia is a frequent infection, especially for people with chronic diseases and for old people. It is one of the most common causes of mortality related to infectious diseases (the first in EEUU).

There are a lot of microorganisms causing pneumonia. The diagnosis depends on the identification of the microorganism causing pneumonia. Nowadays, with the available diagnostic methodology, it is still very difficult to determine which of the microorganisms is the infecting agent in a particular pneumonia case. The research focused to determine which are the microorganisms causing pneumonia only succeeds in the 50% of the cases [1, 5]. Errors in diagnosis can be fatal, for instance, to diagnose *pneumococcal pneumonia* in a patient with *legionella pneumonia* may produce the death of the patient by the delay of an adequate treatment.

Despite the uncertainty of the diagnosis, a treatment has to be speedily administrated to avoid a negative evolution of the severity of the illness or in some cases the death.

In our approach we make two main assumptions:

1) *Existence of a previous diagnosis:* Normally a pneumonia is caused by only one microorganism, but symptoms and signs are not specific enough to determine which one. Diagnosis usually gives evidence for two or three microorganisms possibly causing pneumonia. We will assume in this work that such diagnosis already exists. It can be obtained from another knowledge-based system such as *Pneumon–IA* [6].

2) *Independence of treatments:* We can independently find the best treatment for every microorganism appearing in a diagnosis. Moreover, these treatments can be combined to give a treatment covering all the possible causes. By "covering" we mean that a treatment is specific for a particular microorganism, in other words, it "attacks" the microorganism.

Besides the uncertainty of the diagnosis, data about the patient is in many cases also uncertain and incomplete.

In the subsequent sections we will describe in detail how a solution to this problem has been implemented. We begin with a conceptual structure of the problem. After that we deal with the main ideas used in the real implementation of *Terap–IA*, a knowledge-based system for the automatic generation of treatments. Finally the conclusions and results obtained up to date are presented.

2 Structure of the problem

In Figure 1 we can see the general conceptual structure of *Terap–IA*. Briefly, a treatment for pneumonia is a set of upmost three antibiotics. The antibiotics

*This research has been supported by the Spanish Comisión Interministerial de Ciencia y Tecnología (CICYT) through the project SMASH (TIC96–1038–C04–01).

of the treatment must be possible to administrate to the patient and, cover all the diagnosed microorganisms —because one of them is possibly the cause of pneumonia. The conceptual structure has four parts: first, independently of the diagnosis the system determines all the antibiotics that is possible to administrate to the patient; second, from these antibiotics, the system finds which are the most useful for each of the microorganisms of the diagnosis; third, these treatments are combined producing a treatment covering all the microorganisms of the diagnosis for that concrete patient; finally, the antibiotic combinations are refined with other criteria.

The following sections describe how have we mapped this structure into an operational knowledge-based system developed on top of **Milord II**.

2.1 Ontology

The concepts managed in *Terap–IA* are those related to the pharmacological knowledge about pneumonia treatments and those representing the clinical condition of the patient. The goal of the system is to find the best combination of antibiotics to treat a patient with pneumonia. During the treatment generation process we "weigh" concepts, for instance, the antibiotic adequacy, to give a ranking of final results hence, the clinician can decide to choose an option in the ranking taking into account his preferences.

Pharmacological knowledge *Antibiotics* are pharmacological products for infections' treatment. Antibiotics belong to different pharmacological groups, sharing different properties. For instance, all antibiotics of the same group have a similar spectrum of activity, that is, the set of microorganisms covered by the antibiotics is similar. On the other hand, antibiotics belonging to the same group have different particular characteristics such as cost, administration route, interaction with other drugs, and so on. For instance, *amoxicillin* and *procaine-penicillin G* belong to the group *penicillins* and they cover *gram-positive bacteries*, but *amoxicillin* is orally administrated and *procaine-penicillin G* is intramuscularly administrated.

We consider an *antibiotic treatment* for a microorganism as a set of *antibiotics*, normally one and, occasionally two. An *antibiotic combination* is the result of covering more than one microorganism. It is a set of one[1], two, or exceptionally three antibiotics. The system represents knowledge about all these aspects of antibiotics.

Patient data From the pharmacological knowledge explained above and the particular data of a patient, the system is able to deduce the best treatment for that patient. The most important patient data is the diagnosis[2] consisting of a list of the microorganisms possibly causing the pneumonia (normally upmost three microorganisms).

Other relevant data of the patient is relative to clinical history, physical examination, laboratory test and chest radiograph data, renal function, severity of illness, allergic reactions, genetics alterations and so on.

For instance, we have to take into account the intake of other drugs by the patient to avoid undesired *interactions* among them. Another example is that the administration of antibiotics before the onset of pneumonia can produce that the microorganism producing pneumonia develops *resistance* to some antibiotics.

2.2 Tasks

We have modelled the solution to the problem as a series of tasks that are performed in sequence (see the boxes with text in bold in Figure 1).

A) The starting point of the system is to consider all the antibiotic groups. These groups are *sieved* considering relevant data of the patient, that is, the task consists in finding the degree of adequacy of every group of antibiotics to that patient. It is performed taking into account data about pregnancy, allergic reactions to antibiotics, renal diseases or genetic alterations. The result of this task is the adequacy of every group to a concrete patient. It is independent of the disease we want to treat.

B) Then, taking into account the diagnosis of the patient, we dynamically generate (by selection) a set of treatment tasks, one for each microorganism that possibly caused the pneumonia.

C) Taking into account the adequacy of every group of antibiotics (task **A**) for the patient, every treatment task obtains the best set of antibiotics, for the treatment of the microorganism it is specialised on,

[1] Some antibiotics cover more than one microorganism.

[2] In [6] the interested reader can find an expert system for the diagnosis of pneumonia written in a previous version of **Milord II**.

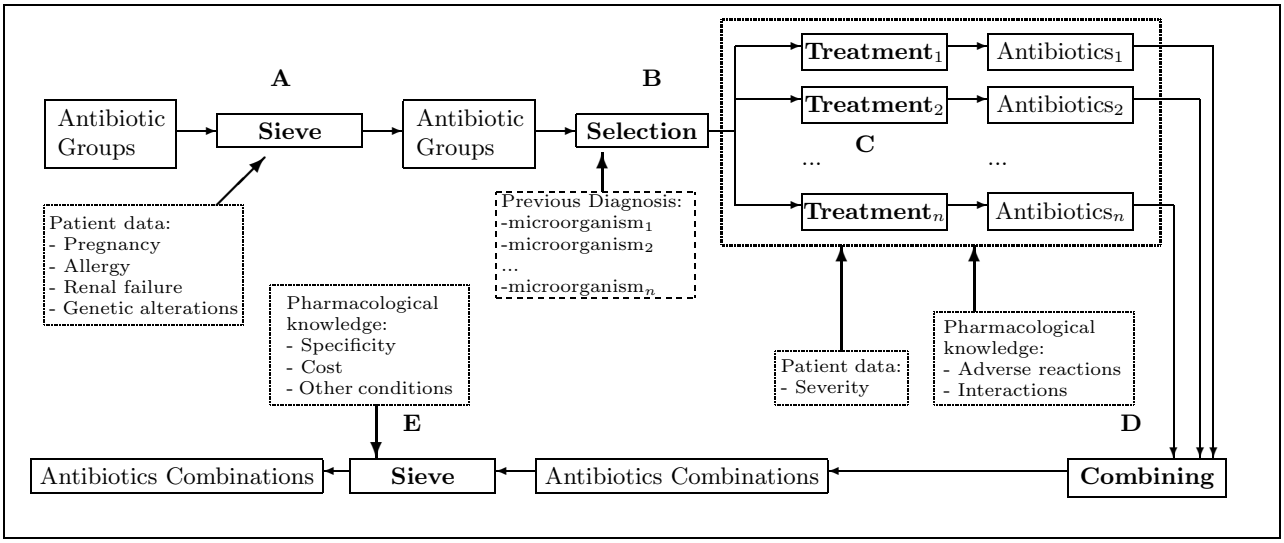

Figure 1: Architecture of *Terap–IA* application.

that belongs to those sieved groups. A treatment task uses data about the patient, the more important is the patient's severity of illness, and pharmacological knowledge about the interactions between drugs, and their contrary effects (previously known adverse reactions to some antibiotics in a concrete patient).

D) The sets of antibiotics adequate for each microorganism are then *combined*. Combinations are produced by means of several criteria. For instance: do not combine antibiotics of the same group, nor those of the same spectrum (equivalent antimicroorganisms activity) nor with different administration route; we prefer monotherapy (one antibiotic) to combinations.

E) Finally, a new *sieve* is applied, now over the ordered set of antibiotics combinations taking into account the specificity and cost, giving as final result an ordered set of combinations that are the most adequate for the treatment of the patient.

The final result of this process is an adequacy-ordered set of treatments (combinations of antibiotics) for a particular pneumonia case of a concrete patient.

3 *Terap–IA*

Terap–IA has been implemented using the **Milord II** language. In this section we present a brief schema of *Terap–IA*'s architecture and which are the decisions in the representation that fulfils the schema presented in Section 2.

Milord II is a modular language for knowledge-based systems. In [2, 3, 4] the interested reader can find a complete description of the language and its logical semantics. We introduce it progressively by means of examples from *Terap–IA*.

3.1 Uncertainty

The approximate reasoning capability of **Milord II** is based on a family of finitely-valued logics which are local to each module and defined as an algebra of truth-values. This allows the system to use a degree of truthness for the concepts involved in the system, then giving graduated results in function of the inherent uncertainty of the data and knowledge involved in the treatment of pneumonia.

The user-defined logic of a module is composed by the declarations of: 1) an ordered set of linguistics terms representing truthness degrees between *true* and *false* and, 2) a *conjunction* operator.

In *Terap–IA* we have used the same set of eight linguistic terms and the same connective modelling in all modules. The terms are the following: *impossible*, *very few possible*, *few possible*, *slightly possible*, *possible*, *quite possible*, *very possible* and *definite*, where *impossible* stands for the boolean *false* and *definite* for the boolean *true*.

Table 1 represents the conjunction operator used in *Terap–IA*, where the linguistic terms are abbrevi-

ated.

	i	vf	fp	sp	p	qp	vp	d
i	i	i	i	i	i	i	i	i
vf	i	vf	vf	vf	vf	vf	vf	vf
fp	i	vf	vf	fp	fp	fp	fp	fp
sp	i	vf	fp	sp	sp	sp	sp	sp
p	i	vf	fp	sp	sp	sp	p	p
qp	i	vf	fp	sp	sp	p	qp	qp
vp	i	vf	fp	sp	p	qp	vp	vp
d	i	vf	fp	sp	p	qp	vp	d

Table 1: Conjunction table for *Terap–IA*.

The many-valued logic of a module is completely determined as soon as the set of linguistic terms and the conjunction operator are chosen. So, varying these two features we may generate different multiple-valued logics. In *Terap–IA* we use the same logic for all modules.

3.2 Propositions and variables

Propositions and variables are the simplest knowledge representation units in **Milord II**. They are named structures that represent the concepts dealt within a module. Their declaration is made by binding an atomic name (identifier) with a set of attributes. The attributes may be a long name, the type, relations with other propositions or variables, a question and so on. Propositions and variables declarations can contain user-defined relations with other propositions and variables. We use this possibility in this application, to model, for instance, relations between antibiotics.

The type is the only attribute that is mandatory in propositions and variables declarations and determines the set of allowed values a proposition or a variable can take, apart from the special value *unknown*, meaning ignorance of the value.

There are three types of propositions, *boolean*, *many-valued* and *fuzzy*; and three types of variables, *numerical*, *linguistic* and *set*. Only *many-valued* propositions or *set* variables can appear in the conclusions of rules, as well as in premises of rules. *Boolean* and *fuzzy* propositions, and *numerical* and *linguistic* variables, are not deduced by the system, they are asked to the user and used only in the premises of rules.

Boolean propositions: They represent concepts which can only be evaluated as either *false* or *true*. In *Terap–IA* most concepts related to antecedents of the patient are boolean propositions, for instance, the knowledge of whether the patient has or not *previous adverse reactions* to some antibiotics, or whether he is affected or not by a *chronic disease*.

Numerical variables: The value of a numerical variable is a real number. For instance, the presence in the blood of a patient of *creatinine* (blood substance which can reach toxic levels with a kidney malfunction) or the *number of leukocytes* can be numerical variables.

Fuzzy propositions: In some cases we need to deal with vague concepts. For instance, taking the example above, we may be interested in the truthness of the *presence of creatinine* useful to support a renal failure of the patient, instead of the numerical value of *creatinine*. Vagueness of concepts can be quantified by the degree of membership of a numerical value to a fuzzy set, so, the *presence of creatinine* can be modelled by giving a fuzzy set (see Figure 2) that takes as argument the numerical value of *creatinine*.

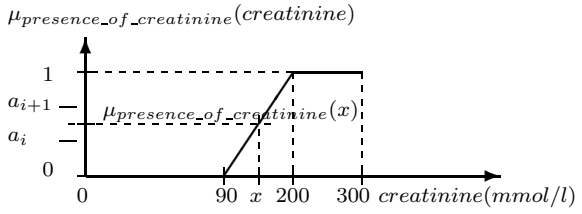

Figure 2: Fuzzy set representing the concept *creatinine*.

The value of a fuzzy proposition is obtained by the application of the fuzzy membership function to the value of a numerical variable obtained from the user of the system. This application returns a number in the interval $[0, 1]$. The final answer is the minimum interval of linguistic terms[3] containing that number.

Linguistic variables: Their values belong to a user-defined finite set of linguistic values. Similarly to the case of fuzzy propositions we can declare a linguistic variable by giving, for every linguistic value, a fuzzy set with respect to a numerical variable.

In Figure 3 we can see a representation of the concept *state of white blood cells* (*swbc* for short) by means of three fuzzy sets (linguistic values), *leukopenia*, *normal* and *leukocytosis*. Given a numerical value for *leukocytes* the system can calculate the value for each linguistic value —the value for *swbc is leukopenia*,

[3]For this purpose we consider the set of linguistic terms to be uniformly distributed in the interval $[0, 1]$.

swbc is normal and *swbc is leukocytosis*— by applying the corresponding fuzzy sets.

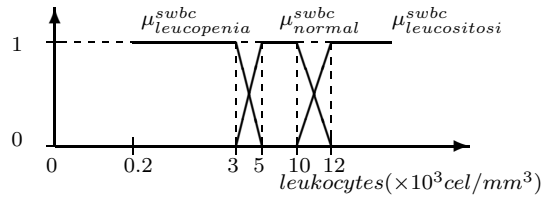

Figure 3: Linguistic variable representing the concept *leukocytes*.

Many-valued propositions: The concepts represented as many-valued propositions are those whose truth may be graded. That is the case of most deduced concepts, for instance, the degree of *severity of illness of the patient* or the degree of *resistance of pneumococci to penicillin*.

Set variables: They are conjunctive fuzzy sets. An example is the concept *allergic reactions*. It is a set which domain is the set of possible allergic reactions of the patient. If a patient has only an allergy to penicillin, the set variable *allergic reactions* is a set with value *true* for penicillin and *false* for the other allergic reactions of the domain.

Given different set variables we could be interested in applying fuzzy set relations and operations. For instance, we can compare different sets by computing, its intersection degree or inclusion degree. For example, we can say in which degree the set variable *allergic reactions* intersects with the crisp set with elements *penicillin* and *macrolides*, that is, which is the degree of the presence of *penicillin* and *macrolides* in the set allergies.

3.3 Rules

In **Milord II**, a rule is composed of an identifier, a premise (a conjunction of conditions), a conclusion, and a truth-value. In the case of conditions containing variables, the language is provided with a set of predefined predicates that apply on them to produce as result intervals of truth values.

Premises of rules are conjunctions of elemental conditions either in affirmative or in negative form. Conditions can contain propositions directly.

For instance, for the fuzzy proposition *presence of creatinine* defined above, we can say *If presence of creatinine then renal failure is definite*. Another example using many-valued propositions is *If a penicillin can be administrated to a patient and his situation is severe then it is very few possible to administrate ampicillin*, in **Milord II** syntax: `If penicillin and severe then conclude ampicillin is vf`.

A numerical expression is composed by numbers, numerical variables and arithmetic operations. For instance, *If respiratory frequency is greater than 30 breaths/minute then we can conclude that tachypnea is definite*.

We can build set expressions in a similar way as numerical expressions. These expressions return degrees of inclusion, intersection and equality between two fuzzy sets. For instance, *If the patient has allergic reaction to penicillin then he is also allergic to cephalosporins and cabapenems*. In this case *allergic to penicillin* means that the set variable *allergies* has intersection with the crisp set $\{penicillin\}$.

For an example of linguistic variable we can use the previously defined linguistic variable *state of white blood cells* to say: *If state of white blood cells is leukopenia then there are analytical evidence of severity of pneumonia*.

Conclusions of rules are simpler than conditions. Conclusions may appear on affirmative or on negative form. Only many-valued propositions and set variables can be used as conclusions in rules. In fact we have seen examples of conclusions if the examples above. For instance the conclusion *tachypnea is definite* is about a many-valued proposition and, *the patient has allergic reaction to cephalosporins and cabapenems* about a set variable.

3.4 Modular structure

The structural construct of **Milord II** is the *module*. A program consists of a set of modules that can recursively contain other modules, then forming a hierarchy.

A module declaration can be clustered in the next sets of components: *hierarchy*, *interface*, *deductive knowledge* and *control knowledge*.

- *Hierarchy:* Is a set of submodule declarations.
- *Interface:* It has two components, the import and the export interface. That is, which propositions and variables may be asked to the user (import) and those whose value is computed by the module (export).

- *Deductive knowledge:* It contains a dictionary declaration, that is, the set of propositions and variables (those belonging to the interface of the module and other intermediate ones) and their attributes; a set of rule declarations, that is, a set of propositional weighted rules; and an inference system declaration, that is, the local logic of the module.
- *Control knowledge:* It is expressed in a metalanguage which acts by reflection over the deductive knowledge and the hierarchy of submodules.

The implementation of the conceptual scheme of Figure 1 is made by programming several **Milord II** modules (about one hundred). It is not possible here to explain in detail every module of the system, thought, we will give some idea of their commonalities by explaining groups of modules.

Pharmacological model modules These modules contain the knowledge about antibiotics for the *Terap–IA* domain. We represent each antibiotic as a concept for which we declare to which *pharmacological group* it belongs; which is the *administration route* of that antibiotic (oral or parenteral); the possible *interactions* with other drugs administrated to the patient and which are the antibiotics with the same *activity*.

For instance, *erythromycin* is an antibiotic that belongs to the group of *macrolides*, it interacts with another drug, *teophylline*, and it has the same sensibility than the antibiotic *doxycycline*. The administration source of this antibiotic can be both: oral or parenteral.

This information will be used by modules that reason about antibiotics. Following the example above it is not adequate to administrate *erythromycin* to a patient already taking *teophylline*, it is better to administrate an antibiotic without interaction with *teophylline*, for instance, *roxithromycin*.

All the concepts above (antibiotics, pharmacological groups of antibiotics, other drugs, administration source) are declared into the pharmacological model modules. Every propositions or variable declaration contains relations with other propositions and variables, for instance, we declare that the antibiotic *erythromycin* belongs to the group of *macrolides*.

Data acquisition modules These modules are the responsible of gathering the patient data that the expert has considered to be relevant for a correct treatment determination. For instance the *number of leukocytes* in the blood of the patient is an important laboratory test data. It is useful to determine if there is *penicillin-resistant* to *pneumococci* microorganism, that is, whether *penicillin* is adequate to treat *pneumococci* or not.

A data acquisition module usually deduces values for concepts different from those acquired from the patient. For instance, one of them deduces *penicillin-resistant* from the *number of leukocytes* (we consider that there is *penicillin-resistant* when the *number of leukocytes* is less that 5000 cells per a mm^3.

Many concepts in the domain are vague. In some cases quantitative data is translated into a qualitative one by means of fuzzy sets. For instance, it is easier for the expert to reason about the concept *the state of white blood cells is normal* than *the number of leukocytes is 7000 cel/mm^3*. This qualitative abstraction allows the expert to say, for example, that there is *penicillin-resistance of pneumococci* when the *state of white blood cells is leukopenia*. You can find examples of qualitative abstractions in Section 3.2.

Examples of this type of modules are: the related to the *clinical history* of the patient; his *physical examination*; his *laboratory test and chest radiograph data*; the possible *complications* (for instance, associated infectious diseases as *meningitis* or *arthritis*) and *the severity of illness*.

Sieve modules group These modules, independently of the microorganism we want to treat, modify the current antibiotic treatment by eliminating the antibiotics belonging to concrete groups of antibiotics of the list of possible treatments for a patient. For instance, it is not possible to administrate antibiotics of the group *fluoroquinolones* to a pregnant patient, because it would affect the fetus.

These modules use the pharmacological model modules and acquisition data modules to determine which are the groups of antibiotics that is possible to administrate to a given patient.

The modules of this group are: *pregnancy*; *allergies*; *renal failure*; and *genetic conditions*. Each module uses data about the patient (from the data acquisition modules) and exports groups of antibiotics. The truth-value of every group of antibiotics is weighted taking into account the adequacy of that group of antibiotics for the given patient. For instance, if the patient has allergic reaction to penicillin, the module of allergies gives to the penicillin group the truth-value *false*.

Operationally we start from a truth-value of *definite* for each group of antibiotics, and then every sieve

module sequentially "sieves" them, keeping the same truth-value if there is no reason against the group, or decrementing it if there is. When the truth-value is *false* we consider that the group has been eliminated from the list of potential groups.

Microorganisms modules groups There are twenty-two groups of microorganism modules, one group for each microorganism treated by *Terap–IA*. Each group of microorganism modules uses the output obtained by the pharmacological model modules, data acquisition modules (specially those about severity of the illness, adverse reactions and interactions) and by the sieve modules group. They deduce which antibiotics to use to treat a microorganism, giving a truth-value for each antibiotic. The truth-value of an antibiotic is obtained taking into account the truth-value of its group (obtained by the sieve modules) and data about the patient.

Given a *diagnosis* (one, two or three possible microorganisms) the system only executes the microorganisms' module groups corresponding to the microorganisms in the diagnosis.

Combination modules group These modules combine the results of the microorganisms modules. The modules providing input to this task export truth-values for antibiotics. Truth-values represent the adequacy of antibiotics for treating the microorganisms. The results of these modules are weitghed antibiotic combinations.

There are several criteria to combine the antibiotics, implemented in these modules. The combinations are of one (monotherapy), two or upmost three antibiotics. Some of the criteria are:

- Combinations are never made with antibiotics belonging to the same group.
- Combinations are never made with antibiotics that have the same sensibility, that is, antibiotics that cover the same microorganisms.
- Combinations are never made with antibiotics that have different administration route (oral or parenteral).
- Given the same adequacy (truth-value) we prefer a monotherapy than a combination with two or three antibiotics.

Sieve combinations modules group The combinations of antibiotics are "sieved" taking into account other criteria, as specificity or cost.

Finally, the system gives to the user the ordered set of antibiotic combinations possible to administrate to the concrete patient.

4 Conclusions and Results

We have presented the *Terap–IA* knowledge-based system. The application takes into account the findings of the patient, knowledge about antibiotics and the diagnosis to find the best treatment. We also have introduced **Milord II** as an appropriate tool and language to develop knowledge-based systems[4].

The expert has developed a partial validation of the system contrasting real world cases of antibiotic treatment of pneumonia with the answers of *Terap–IA*. The treatment for each previous diagnosis has been verified separately, producing hopeful results.

The final validation will be made by giving fifty real world cases of pneumonia to four human experts and *Terap–IA*, and comparing the results.

References

[1] J. Almirall et al. Incidence of community-adquired pneumonia and chamydia pneumoniae infection: a prospective multicentre study. *Eur Respir*, 6:14–18, 1993.

[2] J. Puyol-Gruart, L. Godo, and C. Sierra. Specialisation calculus and communication. *International Journal of Approximate Reasoning (IJAR)*, 18(1/2):107–130, 1998.

[3] J. Puyol-Gruart and C. Sierra. Milord II: a language description. *Mathware and Soft Computing*, 4(3):299–338, 1997.

[4] Josep Puyol-Gruart. *MILORD II: A Language for Knowledge-Based Systems*, volume 1 of *Monografies del IIIA*. IIIA–CSIC, 1996. ISBN: 84–00–07499–8.

[5] A. Torres, J. Serra-Batlles, A. Ferrer, and P. Jimenez. Severe community-adquired pneumonia. etiology prognosis and treatment. *Am Rev Respir Dis*, (144):312–318, 1991.

[6] A. Verdaguer. *Pneumon–IA: Desenvolupament i validació d'un sistema expert d'ajuda*

[4] This software is available for research and educational purposes. You can find the last version and more information on **Milord II** at http://www.iiia.csic.es/~milord.

al diagnòstic mèdic. PhD thesis, Universitat Autònoma de Barcelona, 1989.