ABSTRACT
Automated reasoning techniques are gaining popularity in administrative and legal fields. In particular, when a huge number of heterogeneous reasoning rules must be taken into account, they allow both ensuring the coherence of the system and making the decision process equitable and more efficient. The National Family Benefits Fund of the Grand-Duchy of Luxembourg is responsible for the attribution of allowances to more than 160,000 individuals whose cases, due to the peculiarity of the local economy based on foreign laborers, and given the European and bilateral agreements between countries, pertain to different legislations. Accordingly, the Fund has decided the development of a customized decision support system, named Cadral, for processing allowance applications. The knowledge of the system is contained into decision rules that express administrative procedures used for processing the applications. An intuitive formalism has been set up for the rules, along with an easy-to-use editor, so that a non-computer specialized user can update the system, according to the evolution of the law. The inference engine is based on the Soar forward-chaining architecture. In addition, links to a legal database, used in connection with the reasoning trace of the system, allows exhibiting a legal justification of the resulting decisions.

1. INTRODUCTION
As decision support systems become widely used to disentangle intricate situations and perform sophisticated analysis, the management and visualization of their knowledge form the key points to ensure the system’s proper functioning and keep an intuitive view of its expected behavior. The interests of these reasoning mechanisms are in fact to provide a modeling of the reasoning, as well as a means to perform it in a standardized, and justified way [1]. Decision support systems have in fact taken a significant importance to solve problems that can be formalized through inference rules and combinational cases, notably for identification, planning, optimizing and decision making tasks [2], [3]. These techniques also gain ground in the judicial and administrative domains, especially for the efficient and equitable handling of problems consisting in opening rights or according some pre-defined status.

Cadral (standing for CALcul du DRoit ALlocataire or Beneficiary Allowance Calculation) is a customizable automated tool developed for processing family allowances applications at the National Family Benefits Fund of the Grand-Duchy of Luxembourg (hereafter referred to as the Fund). The daily operating of the Fund consists in the attribution of family allowances and parental leave for more than 160,000 individuals, including 100,000 families and one third of cross-border workers. Handled administrative procedures are therefore very complex due to the local open-economy where individual cases pertain to different national, supra-national and bilateral legal frameworks. Therefore, a tool for the automatic treatment of the demands has long been waited for by the Fund, as being the only means to cope with the continuous increasing of the work, in amount and complexity, due to the demographic and economic expansion of the country.

The procedural knowledge used by Cadral is accessed through 2 different views. In the analytical view, the knowledge is modeled on elementary if-then rules, which are processed by a resolution engine written in the Soar architecture. The synthetic view offers a pictorial representation of all the knowledge, and in particular, shows the inter-dependence of the rules and their legal references. In this architecture the legal data (as such subject to frequent evolution, dismissal, or addition) forming the system's knowledge can be easily updated by a non-computer specialized user. In addition to allowing an efficient processing, the system checks the coherence of the legal framework and guarantees the equitability of individual decisions.
2. THE CADRAL MODEL

Cadral is designed as an inference system, in which rules implement the administrative procedures used to process the applications filled by the beneficiaries to receive the allowances. In the forthcoming version, the system replies on the acceptance of an application for allowances. The information and material supplied through the application is formalized and passed as inputs to the engine processing the rules. Upon completion of the inferences, the demand is accepted, rejected, or the system asks for more information or additional documents. An extension performing the automatic calculation of the allowances is targeted for the subsequent versions of the system.

The starting point for building the system is the national legislation regarding family allowances. It consists however not in a monolithic structure, but forms a constellation of national laws, European Communities decisions, and international agreements.

Modeling juridical texts into logical formalism has been proposed, in order to directly apply inference mechanisms in the context of the law [4], [5]. Several formalisms and resolution algorithms are available and well-tried, such as the first-order and backward-chaining architecture of Prolog, the modal or deontic logics [6]. However, the heterogeneous nature of the texts (fuzzy formalism of general agreements, vs. precision of national administrative code), and the numerous implicit definitions that are used (e.g. certificate validity, prenatal, post-natal allowance...) make the translation into a formal and univocal computer language a very long and minute challenge.

Nevertheless, in the daily work of the Fund, the legislation is not constantly referred to. Operators have in their brain the condensed information that are relevant for most of the cases, and refer to the law only when necessitated by some subtlety of a treatment. Accordingly, instead of the whole mathematical modeling of the law, we decided to concentrate on the explicit drawing of the mental procedures that governs the processing of the applications, and the relations between these procedures to the legislation. The operating knowledge of the system is therefore a procedural modeling of the legal texts.

An example of a procedure can be: "If a child is going to school, and is younger than 18, the beneficiary is entitled to receive the education allowance". Such a procedure is modeled in the shape of a multi-valued acyclic n-are graph, with nodes representing a factual state (e.g. child going to school) used as a condition, and the edges denoting the necessary steps (e.g. showing a school certificate) to enter the state.

Moreover, we ensure that a state is always unique in the graph (there are not 2 nodes with the same label), though it is fully possible to go to the same state by different ways. Let us consider the text of one law article. This text consists of several alineas (i.e. paragraphs), each of them associated with a different state. The modeling of the full law article is therefore a procedural graph, where all the states are distinct, and such that we can define an isomorphism, which associates every node in the graph with an alinea. When the procedures are translated into a collection of inference rules for an expert system, such isomorphism is used concurrently with the trace of the inference engine in order to memorize the legal references made during the reasoning performed according to the procedures. The following example illustrates the graph-based modeling of the legal texts. The root node is associated with the alinea 1 of the (simplified) text, the two following node (in going down) with the alineas 2 and 3 respectively. Two additional concluding leaves for accepting or rejecting the allowances show the final status.

![Diagram](image)

*Article: Education allowance*

$1$. An education allowance is granted to support the education of every child.

$2$. The allowance is granted if the child is going into compulsory school and is younger than 18.
predict basic figures according to demographic statistics, planning tasks within the Fund. The board will be able to prevent them to send to the Fund incomplete applications.

On the other side, macro-simulation will ease the view and manage their demands online. A simulation function will enable them to check the acceptability of the demand, know of the amount they are to received, and provide for possible complaints from the beneficiary against the Fund.

When, in the next version, a module is added to process further the applications until the exact calculation of the payment, the system will enable the beneficiaries as well as the Fund to perform simulations by entering customized input data. As regards the beneficiaries, the Fund is getting equipped with an interactive Website to ensure the rigorous handling of the applications. When

3. DISCUSSION

Among several advantages brought by the computerized approach, we will discuss the 3 most important with respect to the missions of the Fund. The first is to deliver the allowances quicker. The electronic processing will also guarantee the full equitability of the process and make the institution more transparent to the public. Last, in the future, the system should be used to enable the beneficiaries and the Fund to perform individuals or macro simulations of the payments.

As for the efficiency, most of the time, automation will perform a full processing of the files. However, in some more complex cases, according to the particularity of the socio-economic situation of the beneficiary, the system will isolate the decisions that can be automatically made, and transmit the file for manual handling for those requiring the intervention of an human operator.

The computerized approach is, moreover, the only means to ensure the rigorous handling of the applications. When processing a file, the inference engine first validates the coherence of the rules applied in the particular case, and then ensures that an identical and consequently equitable answer is given whenever possible. Otherwise, the decision left to the discretionary intervention of a human operator is precisely circumscribed. The trace of the reasoning, along with references to the juridical texts, can be used to justify the decisions with respect to the law, and provide for possible complaints from the beneficiary against the Fund.

When, in the next version, a module is added to process further the applications until the exact calculation of the payment, the system will enable the beneficiaries as well as the Fund to perform simulations by entering customized input data. As regards the beneficiaries, the Fund is getting equipped with an interactive Website to allow them to access their personal files, so as to fill, view and manage their demands online. A simulation function will enable them to check the acceptability of the demand, know of the amount they are to received, and prevent them to send to the Fund incomplete applications.

On the other side, macro-simulation will ease the planning tasks within the Fund. The board will be able to predict basic figures according to demographic statistics, and to appreciate in advance the impact of changes in the legislation for the family allowances.

4. THE EXPERT SYSTEM

Our graph-procedural modeling of the law has guided our choice relative to the technology used to model the procedures and infer with them. The reasoning on a law article consists, indeed, in proceeding from one state to another according to the procedures and the conditions (labeling the edges) that are satisfied. All the paths and all the states must be effectively checked, in order to ensure that no case provided by the law for a given application is left. This consideration orientates our choice towards a rule-based inference system proceeding by parallel forward chaining (contrary to the Prolog like scheme, which proceeds by backward resolution in a depth-first manner).

The Cadral resolution system is developed on top of the Soar IA architecture. Soar is a general-purpose rule language whose inference engine is based on the Rete algorithm [7], and works in a forward-tracking manner: the rules (also called productions) are "if A then B" statements whose meaning is: "if the situation A is satisfied, then create (or produce) the situation B". Soar's purpose is to propose a Unified Theory of the Cognition [8], and the system is backed as the most suitable language for intelligent agents programming [2]. One advantage of Soar is that it can communicate in many ways (through sockets or procedural routines), and allows us to place in the rules requests for information concerning the allowance demand or the beneficiary data.

However, because of its general purpose, writing rules in the Soar language can soon become intricate, relative to the syntax itself as well as to the management of the inference algorithm [9]. Moreover, when slightly modifying the rule base, the behavior of the whole base can change drastically, in a way that the understanding of the change is not intuitive. For this reason, the heart of Cadral is an intermediate language with simplified syntax that is compiled into true Soar formalism. This upper-level layer is designed to provide the user with useful or necessary subroutines in view of the specialized topic of the program. All the subroutines are documented with a stable and proven behavior, corresponding to the expected modeling of the procedures. In particular, the intermediate language implements the required controls on the Soar resolution engine in order to manage the notion of state used within the procedures, and on-the-fly communications.

This high-level language makes also possible to run a graphical editor in which the procedures are directly modeled in the shape of graphs. This editor is based on the tool Jgraphpad [10], developed on Jgraph, the Java...
core graph visualization library that features a powerful array of graphical functionality, and notably allows to easily draw graphs and export them into miscellaneous formats. In Cadral, graphs drawn with Jgraphpad are exported into GXL (Graph eXchange Language) [11], written in the XML standard.

5. PROGRAMMING CADRAL

We can now discuss the implementation of the system and roughly present the high-level language associated with it, which allows any user to program, check and edit the procedural knowledge of Cadral.

All the modules of Cadral run on a Windows station, though they are portable to Unix or Mac OS. For editing the rule base, the user draws the graphs of the procedures within the Jgraphpad editor and save them in GXL files. The compilation from GXL to the intermediate language, and from this one to Soar rules files is accomplished through two compilers written in C++ with the Lex & Yacc [12] packages.

Running Cadral consists then of starting the Soar engine after launching the Soar rules generated files. Several interfaces are provided for running Soar, among which the TSI library, which incorporates Soar into a Tc1/Tk [13] application, and SGIO (Soar General Input Output) [14] C++ libraries to be linked with a C++ executable. We now run Cadral on both interfaces, in executable test modules. All communications (concerning the data concerning the allowance demands, the beneficiary records in the databases, or the result of the resolution) are simulated through standard input / output, and the trace of the resolution is generated in a text file.

We have successfully used the Cadral suite to write the rules for modeling the legislation concerned with the education allowance, and to test the package. This legislation represents however less than 10% of the entire legal framework for family allowances used by the Fund. The procedures are recorded in a graph of 50 states, which produces about 300 lines of intermediate code and 1500 lines of true Soar code.

In the following, we show the a detail of the procedure shown in Figure 1. We use 3 processing states to write the first test procedure: (1) the starting state "test-age" means that we are checking for the age of a child (to deliver education allowance); (2) The state "limite-age-ok" means that the child is not older than the standard limit for delivering the allowance; (3) The state "test-etude" means that the child is above the age limit, and we check if he is following university studies (which can in some cases extend the duration of the allowance period).

The intermediary code directly generated by the graph, is as follows. The rule (A) and (B) correspond respectively to entering the states (2) and (3). They allow inserting communication routines necessitated to check the real value concerning the age. This value is recorded in additional virtual states ("age-ok" or "age-nok").

```
/* (A) test for state 2 */
RULE age-ok /* rule name */
IN enfant.age <= 18 /*input to be tested */
THEN
POST age-ok oui /* virtual state */
END

/* (B) test for state 3 */
RULE age-nok
IN enfant.age > 19
THEN
POST age-nok oui
END

/* (C) entering state 2 */
RULE limite-age-ok
PRE age-okoui /* precedent state */
THEN
FINAL limite-age-ok oui /* resulting state */
END

/* (D) entering state 3 */
RULE test-etudes
PRE age-nok oui
THEN
FINAL test-etudes oui
```

We now run Cadral on both interfaces, in executable test modules. All communications (concerning the data concerning the allowance demands, the beneficiary records in the databases, or the result of the resolution) are simulated through standard input / output, and the trace of the resolution is generated in a text file.

We have successfully used the Cadral suite to write the rules for modeling the legislation concerned with the education allowance, and to test the package. This legislation represents however less than 10% of the entire legal framework for family allowances used by the Fund. The procedures are recorded in a graph of 50 states, which produces about 300 lines of intermediate code and 1500 lines of true Soar code.

In the following, we show the a detail of the procedure shown in Figure 1. We use 3 processing states to write the first test procedure: (1) the starting state "test-age" means that we are checking for the age of a child (to deliver education allowance); (2) The state "limite-age-ok" means that the child is not older than the standard limit for delivering the allowance; (3) The state "test-etude" means that the child is above the age limit, and we check if he is following university studies (which can in some cases extend the duration of the allowance period).

```
/* (A) test for state 2 */
RULE age-ok /* rule name */
IN enfant.age <= 18 /*input to be tested */
THEN
POST age-ok oui /* virtual state */
END

/* (B) test for state 3 */
RULE age-nok
IN enfant.age > 19
THEN
POST age-nok oui
END

/* (C) entering state 2 */
RULE limite-age-ok
PRE age-ok oui /* precedent state */
THEN
FINAL limite-age-ok oui /* resulting state */
END

/* (D) entering state 3 */
RULE test-etudes
PRE age-nok oui
THEN
FINAL test-etudes oui
```

Corresponding Soar code is as follows. The behavior of Soar programs necessitates more introductive explanations, such as given in [9], to be figured out. However, we remark that every rule of the intermediate code generates two Soar rules, the first to propose an "operator" (a function) and the second to apply the operator. Soar can also write in special areas of the memory ("io.input-link" and "io.output-link") in order to communicate with external environment on the fly, during the inference mechanism.

```
sp {
  A-1-propose*age-ok /* rule name */
  (state <s> ^io.input-link <il> /* if part */
   ^age-ok)
  (<il> ^enfant.age <= 18)
  -->
  (</s> ^operator <o> + =) /* then part */
  (</o> ^name age-ok)
}
In order to be integrated in the working infrastructure of the Fund, the system must fulfill basic requirements in terms of interface, usage and usability of the response. First, the system is designed to receive as inputs all the data that can be contained in the application files. An interface is thus maintained with a customizable list of these input data, the list being edited when changes occur in the law concerning allowances requirements. For now, the data are entered manually, but interfaces will be developed for communicating with an OCR-enabled processing module for the paper documents and files, and with the Website for applications filled online.

Still regarding the interface, the system must be able to retrieve all the parameters that are to be taken into account for the processing of the demand, including checking for the presence of required certificates. The data is here dispatched on several databases tables, and special care must be taken to recombine all the information. For example, a cross-border worker is entitled to receive child allowance, but the amount paid by the Fund will deduct the amount of the allowance of the same kind possibly received from the country of residence.

As concerns the practical use, the system is characterized by its flexibility. The knowledge it contains in order to process all the demands is a set of legal texts, and as such, they are subject to frequent evolution, dismissal, or addition. In a similar way, the situation for individual data of the beneficiaries can be either modified in the life, or adjusted according to new juridical status. All these reasoning and demographic data must be stored so as to be easily updated by a non-computer specialist operator of the Fund.

Last, dealing with juridical issues, the answer provided by Cadral must be deterministic, and justified. Due to either intrinsic incoherence or errors in the translation into computerized procedures, the juridical knowledge of the system can be erroneous, or incomplete. In the traditional manual handling, the incoherence is detected and solved by the operator or the hierarchy. With an automatic processing, the situation could lead to unwanted responses. It is therefore necessary to proceed with a set of rules and an inference mechanism that guarantee the exactness of the knowledge and the
uniqueness of the returned decision. Moreover, the reasoning process provided by Soar is in every case traceable. In addition, links are maintained from the rules to a database of legal references, in order to exhibit a juridical justification of the decision.

![Diagram of CADRAL system]

**Figure 3. Cadral Integration**

### 7. CONCLUSION

In the Cadral decision tool, the raw knowledge of the system is defined by a comprehensive set of juridical texts. However, though reference to the law are necessary in some intricate cases and for justifying the results, direct interpretation of the texts can be avoided in most of the cases. We have discussed the feasibility of a method based on the modeling of the legal framework into light procedures. This approach is completed by a graphical visualization of the juridical knowledge and an operational integration in a computerized infrastructure. Future works include primarily the 2 following axes. First, a macro-editing package of the knowledge working from the graphical representation, in order to ease the update of Cadral without rewriting separate rules. Second, a calculation module, in view of both a full automated handling of the benefits applications and the development of micro (for a single user) and macro (for a population) simulation tools.

### REFERENCES


