3
State-of-the-art

For problem solving, formal validation of systems, as well educational purposes, it is convenient the use of proof assistants. There are two main approaches: the automatic proof assistants and the interactive proof assistants.

The intelligence in problem solving is fully concentrated in the specification phase of the set of sentences that represent the problem. From this point, it is possible to submit them to the theorem prover, which would give the derivation that shows the correct answer, if any. It is important to notice that an automatic theorem prover, in most cases, has a set of heuristics, literally speaking, to enhance performance. Theorem proving in propositional logic is a problem which the best known algorithm is exponential time. That is why heuristics are needed. [17] cfr. section 7.1.

For learning and teaching formal reasoning, interactive proof assistants are very useful. Logic students and researchers can use the logic definition language of a proof assistant to try different logics with hands on. Beginners can go forth and backward, step by step, in order to understand the way to apply the inference rules on formulae.

Neither automatic nor interactive proof assistants reach usability ideal. A third approach would be The Saint Thomas Aquinas Machine, which is semi-automatic. An automatic theorem prover, that asks for user interaction only in the most difficult steps. The prover accepts descriptions of logics, being able to support different logics. Each logic description defines in which cases a decision is needed. The machine's architecture supports software agent users making the decisions.

The Saint Thomas Aquinas Machine is a virtual machine that executes code written in an assembly language. Any theorem prover for any logic and deduction system can be written. In order to help developers, an upper level language was defined. A compiler for the upper level language is also provided.

Theorem proving is already an explored area. Some theoretical and practical advances have been made. The logic and formal methods community counts with many theorem provers.

Jape [14] for example, takes a description of a logic as a system of inference rules, and supports the development of proofs in that logic. It has a tactic language which is used to control the display of proofs and to perform simple searches. Significant human effort is needed for simple proofs. Even trivial goals require users to specify rules and tactics. Large proofs are a problem consequently.

Isabelle [13] also accepts the description of different logics and deduction systems. Logics are formulated with a meta logic. Isabelle has a language to instantiate deduction systems. It is a very flexible approach. However, not every deduction system can be naturally instantiated for it. Isabelle is based on tree-structured deductions. Systems like SEQ_0+**WS** (section 2) cannot be instantiated for it. Necessarily, some proofs in Isabelle are much bigger than they should be.

Another popular mechanized formal method is PVS [21]. PVS is distributed with lots of tools to aid the description of deduction systems. Its interactive theorem prover supports the use of several decision procedures. Although its language seems to support many deduction systems; internally, it makes all the proofs as a sequent calculus. This limitation is overcome by the presented approach.

The Cathedral Theorem Proving Platform is graph-based. The formulas and proofs are represented on the machine as graphs. The deduction system is specified as graph transformations. It represents a new way to make automatic theorem provers. With it, it is possible to describe systems that support the new requisites that arouse from theoretical studies (presented in section 2 and [4]).

The presented theorem proving platform supports strategies. Decision making is done through a defined protocol. While proving theorems, the process can be interrupted in order to delegate decisions to an external agent. The proof process is designed as a goal achieving process.