# Logic as Energy: a SAT-based Approach

Priscila M. V. Lima[1], M. Mariela M. Morveli-Espinoza[2,3], and Felipe M. G. França[1,3]

[1] LAM – Computer Architecture and Microelectronics Laboratory
Universidade Federal do Rio de Janeiro, Brazil
priscilamvl@lam.ufrj.br
http://www.lam.ufrj.br/
[2] Departament d'Informàtica
Universitat Autònoma de Barcelona, Spain
fourme@gmail.com
http://upiia.uab.es/
[3] COPPE – Systems Engineering and Computer Science Program
Universidade Federal do Rio de Janeiro, Brazil
{mme,felipe}@cos.ufrj.br
http://www.cos.ufrj.br/~felipe

**Abstract.** This paper presents the implementation of ARQ-PROP II, a limited-depth propositional reasoner, via the compilation of its specification into an exact formulation using the SATyrus platform. SATyrus' compiler takes as input the definition of a problem as a set of pseudo-Boolean constraints and produces, as output, the Energy function of a higher-order artificial neural network. This way, SATisfiability of a formula can be associated to global optima. In the case of ARQ-PROP II, global optima is associated to Resolution-based refutation, in such a way that allows for simplified abduction and prediction to be unified with deduction. Besides experimental results on deduction with ARQ-PROP II, this work also corrects the mapping of SATisfiability into Energy minima originally proposed by Gadi Pinkas.

**Key words:** ARQ-PROP II, higher-order neural networks, propositional reasoner, satisfiability, SATyrus.

## 1 Introduction

Plenty of research has been carried out on how neural networks learn and create implicit knowledge from perceptual experience. On a smaller scale, come the efforts on rule extraction from such knowledge. In the next scale degree, fewer works on how neural networks perform logical reasoning are noticed. However, with very few exceptions [1], not much has been done towards integrating these three approaches. This paper presents the implementation of ARQ-PROP II [8], a neural-based propositional reasoner possessing a writable area so that knowledge coming from outside, e.g., perceptual areas, could be integrated in the reasoning process.

In order to handle ARQ-PROP II's complex architecture, its implementation was realized through the use of the SATyrus' platform [11]. In previous works, it was shown how optimization problems, such as TSP (*Traveling Salesperson Problem*) and graph colouring, could be specified as sets of pseudo-Boolean constraints and easily combined through the concatenation of their respective specifications, plus the addition of other pseudo-Boolean constraints specifying the combination's intentionality [10]. ARQ-PROP II's architecture, declared as pseudo-Boolean constraints, is taken as input to the SATyrus' compiler which produces, as output, an Energy function that can be directly mapped into a higher-order artificial neural network.

SATyrus' compilation process, described in the next section, is based on the mapping of SATisfiability of a formula into global optima of an Energy function, which was originally proposed by Gadi Pinkas [9]. However, such mapping was proven to produce spurious global minima in more complex problems, such as in the case of ARQ-PROP II, and this is corrected in Section 2.2. ARQ-PROP II, presented in Section 3, works by associating global optima to Resolution-based refutation, so that different logical reasoning styles such as abduction, deduction and prediction can be performed in a uniform way. Experimental results from ARQ-PROP II performing deduction are described and discussed in Section 4, followed by our conclusions, presented in the last section.

## 2 SATyrus: a SATisfiability-based architecture for constraint processing

SATyrus platform is basically composed by two modules: a *compiler* and a *solver* [10] [11]. A problem specification is fed to the compiler as a set of pseudo-Boolean constraints, representing both the problem's search space and the cost function, and a *penalty scale* modulating the whole set of constraints. The object code produced by the compiler consists of an Energy function, which can also be seen as a single exact formulation to the problem in question. Global minima of this Energy function, corresponding to the desired set of solutions, are obtained through the use of a solver. The current SATyrus' solver is based on symmetric higher-order neural networks.

### 2.1 SATyrus' language and compiler

Input to SATyrus' compiler consists of a problem specification written in the SATyrus' declarative language. The general structure of a problem specification is divided in four main parts: (i) neural structures, (ii) integrity constraints, (iii) optimality constraints and (iv) a penalty scale associated to the different groups of constraints defined within (ii) and (iii).

Constructs are provided to express the specification of different data structures of binary elements. These structures are, basically, multi-dimensional arrays. The elements of these arrays play the role of propositional variables in the constructs that specify constraints and will be identified as binary neurons

in the neural solver. The replication of such constraints is facilitated by other constructs. The objective function sentences are defined in a similar way and may be read from a file. Also provided are constructs for the association of an identifier to a group of both integrity and optimality constraints, in order to enable the attribution of a same penalty level to them.

### 2.2 Energy function generation

The compiler translates the file containing the problem specification into an intermediate representation composed by one header and a record for each term of the Energy function. Each record has the following information: penalty level, weight, connection arity and list of neighboring neurons. The header provides a table with penalty identifiers and respective values. Only the penalty identifiers and their levels are informed by the user, their values and neurons attributes result from the compilation process.

The association of SATISFIABILITY (SAT) to global minima of a function requires the consideration of the basic mapping of truth values of propositional formulae to the domain $\{0, 1\}$:

$$\begin{aligned} H(true) &= 1 \\ H(false) &= 0 \\ H(\neg p) &= 1 - H(p) \\ H(p \wedge q) &= H(p) \times H(q) \\ H(p \vee q) &= H(p) + H(q) - H(p \wedge q) \end{aligned}$$

If a logical formula is converted to an equivalent in *Conjunctive Normal Form* (CNF), the result being a conjunction $\varphi$ of disjunctions $\varphi_i$, it is possible to associate energy to $H(\neg\varphi)$. Nevertheless, energy calculated in this way would only have two possible values: *one*, meaning solution not found (if the network has not reached global minimum), and *zero* when a model has been found. Intuitively, it would be better to have more "clues", or degrees of "non-satisfiabililty", on whether the network is close to a solution or not. This measure also prevents the Energy equation from having an exponential number of terms which could result from the conversion of the outermost disjunction in the negated formula.

Let $\varphi = \wedge_i \varphi_i$ where $\varphi_i = \vee_j l_{ij}$, and $l_{ij}$ is a literal (either $p_{ij}$ or $\neg p_{ij}$). Therefore $\neg\varphi = \vee_i \neg\varphi_i$ where $\neg\varphi_i = \wedge_j \neg l_{ij}$. Instead of making $E = H(\neg\varphi)$, consider $E = H^*(\neg\varphi) = \sum_i H(\neg\varphi_i)$. So, $E = \sum_i H(\wedge_j \neg l_{ij}) = \sum_i \prod_j H(\neg l_{ij})$, where $H(p)$ will be referred to as $p$. Informally, $E$ counts the number of clauses that are *not satisfied* by the interpretation represented by the network's state.

A simple example demonstrates how SAT can be mapped to EM. Let $\varphi$ be the formula, expressed as a conjunction of clauses:

$$\varphi = (p \vee \neg q) \wedge (p \vee \neg r) \wedge (r)$$

SAT ($\varphi$) can be translated to the minimum of the following energy function:

$$\begin{aligned} E &= H(\neg(p \vee \neg q)) + H(\neg(p \vee \neg r)) + H(\neg r) \\ &= H(\neg p \wedge q) + H(\neg p \wedge r) + H(\neg r) \\ &= (1-p) * q + (1-p) * r + (1-r) \\ &= q - pq - pr + 1 \end{aligned}$$

where $H(p) = p$.

Another source of potentially exponential space cost occurs when a clause $c_i$ is required by the modeling to have a number of literals equal to the size $n$ of the problem. The simplification displayed by function $H^*(\neg\varphi)$ could not be applied in this case. In some situations, however, only one of the disjuncts should be allowed to be true at a time, constituting an exclusive-OR. The definition of a set of so-called *Winner-Takes-All* (WTA) constraints helps to prevent violation of the exclusiveness. This prevention can only be achieved by the attribution to the WTA-constraints of a penalty level higher than $c_i$. Penalty values should be calculated in such a way that no violation of a constraint of level $i$ could be traded for the satisfaction of constraints of lower levels. This can be done automatically, provided that the user informs an upper bound for the optimality constraints, if there are any. It is also worth mentioning that Pinkas' mapping did not consider tackling optimization problems, only logical reasoning ones.

Up to this point, the mapping proposed by Gadi Pinkas has been described. Nevertheless, an important mapping rule has been left unspecified by him, leading to potential spurious global minima. This work proposes the addition of a rule that states that clause $c_i$ should be broken into $n$ singleton clauses $s_{ij}$, $1 \leq j \leq n$. The set $\{s_{ij}\}$ should be associated to a new penalty level, immediately lower than that of the original $c_i$, but still higher than the other lower penalty levels.

### 2.3 SATyrus' neural solver

Once the Energy function is defined, one could apply a number of different solvers in order to find its global minima. In the present work, it is assumed a generalization of Hopfield neural networks [5], where a stochastic behaviour [6] is introduced into its binary neurons, i.e., output $ON = 1$ or $OFF = 0$. It is worth noticing that the symmetric neural network associated to the mapping introduced in the previous subsection may have higher-order connections. This means that the resulting Energy function may have terms with more than two propositional variables, what would imply on having synaptic weights involving more than two neurons each, e.g., neurons $i, j$ and $k$ such that $w_{ijk} = w_{jik} = w_{kij}$. This does not constitute a hindrance as has been demonstrated that, with higher-order connections, Boltzmann Machines still converge to energy minima [4]. Parallel and distributed simulation of networks [2] with higher-order connections can be

done by substituting each higher-order connection by a completely-connected subgraph. Alternatively, [9] converts the higher-order network to a binarily connected one that preserves the order of energy values of the different network states.

## 3 ARQ-PROP II: a goal-driven propositional reasoner

It is possible to use the mechanism described in Section 2 to design a neural engine that is capable of performing propositional Resolution-based reasoning with both complete and incomplete knowledge. The modeling has to define the sets of propositional variables to be associated to binary neurons and a set of constraints that provides the reasoner with the ability to perform sound Resolution steps. Additionally, in order to reason with incomplete knowledge and to have the flexibility that the knowledge base does not be pre-encoded as constraints, the engine has to be able to create new sentences (clauses).

### 3.1 ARQ-PROP II architecture

The data structures of ARQ-PROP II are displayed in Figure 1. The meaning of the states of the elements of the ARQ-PROP II in Figure 1 will be explained in Section 4. The interpretation of ARQ-PROP II structures is the following:

- `IN` $(n \times 1)$: indicates if the line is part of the selected proof or not; $i$ = line number in the proof area;
- `PROOF` $(n \times n \times \{+, -\})$: proof area; $i$ = line number; $j$ = nameprop; $k$ = literal sign;
- `CB-RES-INV` $(n \times 3)$: reason for belonging to the selected proof for a line in the proof area; it can either be an instance of a clause of the Clause Base (`CB`), the result of a resolution step (`RES`), or an invention in the case of reasoning with incomplete knowledge (`INV`); $i$ = line number; $j$ = reason;
- `EMPTY` $(n \times 1)$: indicates whether the line is the empty clause or not; $i$ = line number;
- `CBMAP` $(n \times n)$: maps proof lines to the internal names of clauses of the Clause Base that they derive from; $i$ = line number; $j$ = clause;
- `PARENT` $(n \times n \times \{1, 2\})$: indicates the parents (parent1 or parent2) of a line, resulting from a resolution step, in the proof; $i$ = parent line number; $j$ = line number; $k$ = parent1 or parent2;
- `CANCELED` $(n \times n)$: indicates which proposition has been canceled in the proof lines that result from resolution steps; $i$ = line number; $j$ = nameprop;
- `CLCOMP` $(n \times n \times \{+, -\})$: indicates clause composition for each clause (internal name) of the selected Clause Base; $i$ = clause number; $j$ = nameprop, $k$ = literal sign;
- `ORIG` $(n \times 1)$: indicates that the clause belongs to the original knowledge base; $i$ = clause number.

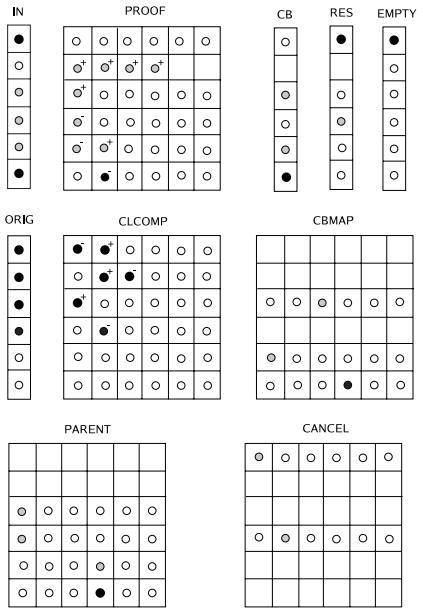


**Fig. 1.** General structures of ARQ-PROP II having a proof depth limit of 6. Final states of neurons after deduction of $\square$ from $\{p \vee \neg q, p \vee \neg r, q\} \cup \{\neg p\}$. The `IN-PROOF` lines compose a refutation for $\neg p$, thus proving $p$: line 1: $\square$ (empty clause), line 3: $q$, line 4: $\neg q$, line 5: $\neg q \vee p$, line 6: $\neg p$ (query).

It is worth pointing out that structures `PROOF`, `CLCOMP` and `PARENT` are tridimensional with $n \times n \times 2$ elements, each. In the first two structures, the third dimension indicates the sign of the propositional literal, while in structure `PARENT`, the third dimension is used to enforce the participation of two different clause instances on a Resolution step. Clauses composition must be indicated explicitly by fixing the values of the nodes of structure `CLCOMP`. The structure `INV` is used to indicate that a proof could be generated provided that sentence(s) were incorporated to the knowledge base.

### 3.2 Set of constraints of ARQ-PROP II

In general, the set of integrity and optimality constraints of ARQ-PROP II must account for the specification of a Resolution step (Resolution step constraints, Parent line constraints and Resolvent composition constraints) and specify the conditions for a line in the `PROOF` area to actually belong to the result of the computation (In-proof constraints). In order to accomplish that, it is necessary to add constraints for the enforcement of clause syntax (Clause instance constraints, Clause syntax constraints, Empty clause constraints, Clause-invention constraints). The whole set of constraints, the detailing of which has been revised in [14], can be informally stated as:

1. Every line of `PROOF` resulting from an inference step, i.e., one that is not a copy of a clause from `CLCOMP`, must have exactly two different parents, which are also lines of `PROOF`;
2. Every line that is a copy of a clause from the Clause Base, `CLCOMP` has no parents;
3. Except from the empty clause, every line of the proof must be a parent of exactly one line in the proof;
4. Every Resolution inference step must have one and only one pair of canceled literals;
5. Apart from the canceled pair of literals, all and only literals of both parents involved in an inference step must be copied to the resulting proof line;
6. Every line that belongs to a proof is either a copy of a clause from the Clause Base or constitutes the result of a Resolution inference step;

Additionally, some WTA conditions have been used to justify the conversion of disjunctions in the middle of constraints to a conjunction of the disjuncts, as explained in Section 2.2, or as a necessary part of the specification of ARQ-PROP II:

7. WTA-1-sign: only one occurrence of propositional symbol (i.e., of its internal name) per line in the proof (applied to `PROOF`);
8. WTA-2-line: only one reason per proof line (applied to `CB-RES-INV`);
9. WTA-3-line: only one clause from the Clause Base (`CLCOMP`) copied per line of `PROOF` area (applied to `CBMAP`);
10. WTA-4-(column, parent1/2): a line can have only one parent1 and only one parent2 (applied to `PARENT`);
11. WTA-5-line: a line in the proof may take part in a resolution step (i.e., be one of the parents of another line) only once (applied to `PARENT`);
12. WTA-6-parent1/2: two different proof lines must be involved in a resolution step (applied to `PARENT`);
13. WTA-7-line: only one pair of literals (i.e., propositional symbol) canceled per line number resulting from a resolution step (applied to `CANCELED`);
14. WTA-8-sign: only one occurrence of propositional symbol (i.e., of its internal name) per clause (applied to `CLCOMP`).

The specification of a pure-deduction reasoner would be complete if constraints 1 to 14 were satisfied. For a simple version of reasoning with incomplete knowledge to take place, the invention of a clause must be penalized, as it is usually more desirable to have a complete deduction of the empty clause. The penalty has to be such that energy of a sentence that belongs to a proof will be smaller if it is possible to choose it from the knowledge base or to generate it from a Resolution step. This is achieved by attributing the lowest penalty level to these constraints, making them optimality constraints.

## 4 Compiling and running ARQ-PROP II with SATyrus

This section presents two experiments exercising ARQ-PROP II on performing deduction over two small clause bases, both having $\neg p$ as query:

$$\Delta_1 = \{p \vee \neg q, p \vee \neg r, q\}$$
$$\Delta_2 = \{p \vee \neg q \vee \neg r, q, r\}$$

As previously shown in Figure 1, the compilation of ARQ-PROP II, assuming a depth limit of 6 in the proof area, resulted in a network having 318 neurons (note that fields `PROOF` and `CLCOMP` have two layers – "+" and "−", as well as field `PARENT` – parent1 and parent2). The clause base $\Delta_1 \cup \{\neg p\}$ was written in the clause base (`CLCOMP`) area. The graphical conventions adopted for the neurons' output are: **black** means the neuron is clamped *ON*; within lines having black neurons, **white** nodes are clamped *OFF*; **grey** neurons are *ON* as a result; within lines having grey neurons, **white** nodes are *OFF* as a result; positions having no output doesn't matter for the current result. Notice that, as illustrated in Figure 1, there are *ON* neurons in the proof area (`PROOF`) representing no input clauses. Such neurons, specially the ones in line 2, should not be considered since their corresponding neurons in field `IN` were set to *OFF*. Those neurons producing *ON* values do not have any influence on the final calculus of the Energy function.

Both experiments were conducted using the same initial temperature $T_i = 10000$, final temperature $T_f = 1$, and a geometrical cooling factor of 0.99. Figure 2 illustrates the behaviour of Energy function in the experiment with ARQ-PROP II performing deduction of $\square$ from $\Delta_1 \cup \{\neg p\}$. In the second experiment, the compilation of ARQ-PROP II assumed a depth limit of 8 in the proof area, resulting in a network having 552 neurons. The clause base $\Delta_2 \cup \{\neg p\}$ was written in the clause base (`CLCOMP`) area and Figure 3 illustrates the behaviour of Energy in this case.

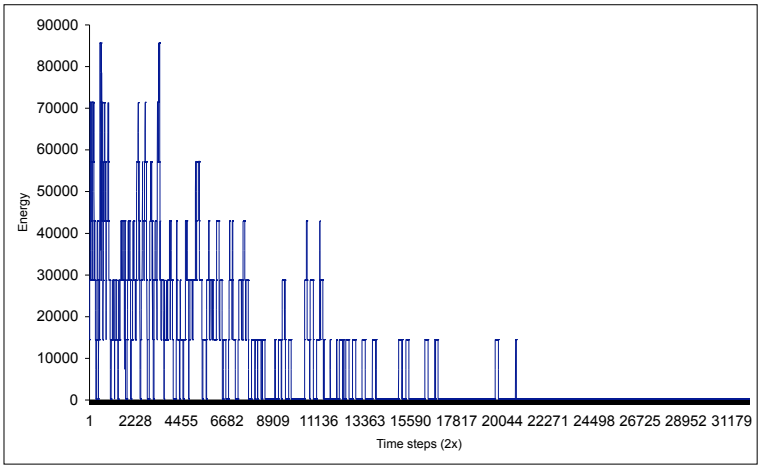


**Fig. 2.** Trajectory of the Energy for deduction of $\square$ from $\Delta_1 \cup \{\neg p\}$; final Energy: 169.

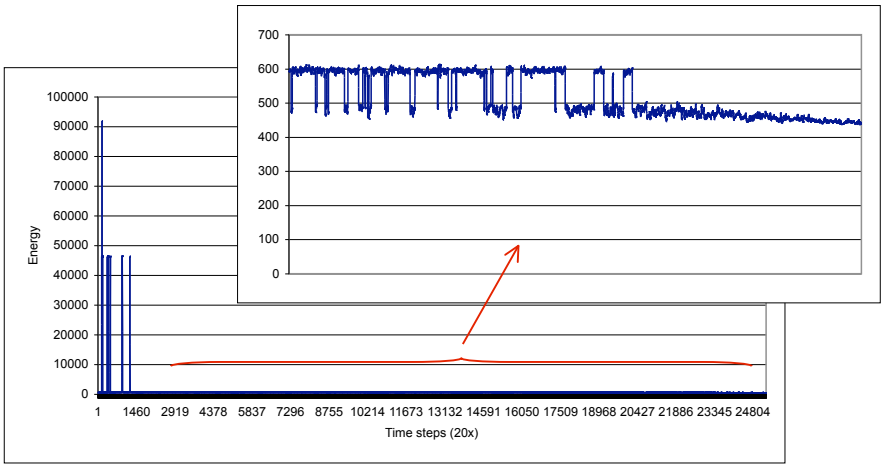


**Fig. 3.** Trajectory of the Energy for deduction of $\square$ from $\Delta_2 \cup \{\neg p\}$; final Energy: 433.

## 5 Conclusion

Apart from the recent resurgence of interest in SATISFIABILITY as a means of overcoming the inherent difficulty of many NP-hard problems [3], interest on Pinkas' original mapping, such as in *Markov Logic Networks* (MLNs) [13], are relatively recent. The main contribution of MLNs lies on the amalgamation of learning by examples with inferencing. On the other hand, the conception of ARQ-PROP II as a generic propositional reasoner is unique when compared to any other purely connectionist approach, since there is no predefined knowledge base involved. Moreover, ARQ-PROP II is able to reason with incomplete knowledge and to create new clauses; both interesting features to be explored in the design of intelligent machines. Besides abduction and prediction, other kinds of reasoning styles, still in the propositional domain, such as planning, are among the next experimentation steps of this research.

Another ongoing work is the full implementation of ARQ-FOL II [7], the First Order Logic (FOL) generalization of ARQ-PROP II. As in ARQ-PROP II, the ARQ-FOL II architecture allows one to set a predefined limit for the proof depth, while working without a predefined knowledge base. Such would be the first neural reasoner performing this logic level. It must be noticed that, although the resulting complexity of both kind of neural architectures are polynomial, time complexity remains exponential. This is reasonable since the reasoners in question do not treat just Horn clauses.

The use of the strategy described in this work for the complete implementation of ARQ-FOL II and other complex problems, such as the energy generation expansion [15] and the molecular geometry reconstruction [12], can only be carried out via an automated process. This is due to the number of terms of the

Energy function generated by the SATyrus compiler, as would be the case of other logical/opmization methods. Adjustment of the search mechanisms of the SATyrus neural solver and exploration of other meta-heuristics constitute ongoing work as well.

**Acknowledgments.** This work was supported by CNPq, CEDERJ (EELA project) and FEST (SCAE project).

## References

1. Aleksander, I.; Evans, R.G.; Sales, N. Towards intentional neural systems: experiments with MAGNUS. *Proc. of the Fourth Int. Conf. on Artificial Neural Networks* (1995) 122–126.
2. Barbosa, V.C, Lima, P.M.V.: On the distributed parallel simulation of Hopfield's neural networks. *Software-Practice and Experience* **20**(10) (1990) 967–983.
3. Dixon, H.E., Ginsberg, M.L., Parkes, A.J.: Generalizing Boolean Satisfiability I: Background and Survey of Existing Work. *J. of Art. Intelligence Research* **21** (2004) 193–243.
4. Geman, S., Geman, D. : Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **PAMI-6** (1984) 721–741.
5. Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. *Proc. of the Nat. Acad. of Sciences* USA **79** (1982) 2554–2558.
6. Kirkpatrick, S., Gellat Jr., C.D., Vecchi, M.P.: Optimization via Simulated Annealing. *Science* **220** (1983) 671–680.
7. Lima, P.M.V.: *Resolution-Based Inference on Artificial Neural Networks.* Ph.D. Thesis, Department of Computing. Imperial College London, UK (2000).
8. Lima, P.M.V.: A Goal-Driven Neural Propositional Interpreter. *International Journal of Neural Systems* **11** (2001) 311–322.
9. Pinkas, G.: *Logical Inference in Symmetric Neural Networks.* D.Sc. Thesis, Sever Institute of Technology, Washington University, Saint Louis, USA (1992).
10. Lima, P.M.V., Pereira, G.C., Morveli-Espinoza, M.M.M. and França, F.M.G.: Mapping and Combining Combinatorial Problems into Energy Landscapes via Pseudo-Boolean Constraints. *LNCS* **3704** (Proc. of BV&AI 2005) (2005) 308–317.
11. Lima, P.M.V., Pereira, G.C., Morveli-Espinoza, M.M.M. and França, F.M.G.: SATyrus: A SAT-based Neuro-Symbolic Architecture for Constraint Processing. *Proc. of the Fifth Int. Conf. on Hybrid Intelligent Systems* (2005) 137–142.
12. Lima, P. M. V. ; Pereira, G. C. ; Morveli-Espinoza, M. M. M. ; França, F. M. G. ; Lavor, C. C.: Mapping Molecular Geometry Problems into Pseudo-Boolean Constraints. *Proc. of the Int. Work. on Genomic Databases – IWGD05* (`http://www.biowebdb.org/iwgd05/proceedings/index.html`) (2005)
13. Richardson, M., Domingos P.: Markov Logic Networks. *Machine Learning* **62** (2006) 107–136.
14. Morveli-Espinoza, M.M.M.: *Compiling Problems Resolution to Energy Minimization.* M.Sc. dissert., COPPE/PESC, Universidade Federal do Rio de Janeiro (2006).
15. Silva, E. F. ; Lima, P. M. V. ; Diacovo, R. ; França, F. M. G.: Aggregating energy scenarios using the SATyrus neuro-symbolic tool. *Abstracts of the 19th Int. Symp. on Mathematical Programming* (2006) 146–146.