# Computational Modeling of Abductive Reasoning:

## an experimental study employing the CHR language in an educational context

Osvaldo Luiz de Oliveira
Computer Science Department
FACCAMP
Campo Limpo Paulista, São Paulo, Brazil
osvaldo@faccamp.br

Ricardo José Martins
Federal Institute of Education, Science and Technology
FACCAMP & IFSULDEMINAS
Muzambinho, Minas Gerais, Brazil
ricardo.martins@muz.ifsuldeminas.edu.br

***Abstract*—Computational modeling is the process of programming models of machines, circuits, buildings, phenomena etc. This activity has been widely researched and used in practice at different levels of formal education for a student to learn concepts and techniques on engineering, natural sciences, economics, demography, geography etc. via model programming. Abduction is a type of reasoning that aims to formulate hypotheses to explain observed facts via a theory. Computational systems to program abductive reasoning, such as the Constraint Handling Rules (CHR) language, have been developed and researched; however, they have not been studied in an educational context. This work proposes and experimentally investigates the use of the CHR language to create an educational environment for abductive educational computational modeling.**

***Keywords—computational modeling; constructivism; abduction; abductive reasoning.***

## I. INTRODUCTION

Computational modeling refers to the development of programs to simulate real or abstract phenomena. In the education field, modeling has been used as a didactic approach to implement a constructivist educational paradigm in which the student learns via activities such as the development of computational models of machines, circuits, buildings, phenomena, and so on. Basically, an environment for computational modeling offers a programming language to the student and creates a cycle in which, for example, the student develops a program that simulates a machine, runs the program, interprets the results, and if the results are inconsistent with the machine, the student updates the program developed. This process in which programs are developed and updated allows students to improve the model of a machine and thus to also improve his knowledge.

The Constraint Handling Rules (CHR) language has been widely used in abductive reasoning programming to solve several problems, including design, planning, and fault diagnosis [7]. It allows writing programs to compute abductive reasoning, for example, on phenomena or on the operation of machines.

Although abductive reasoning [21] programming using the CHR language can be used in an educational modeling context, no studies have been conducted on this topic. This work proposes using the CHR language for abductive reasoning programming in educational computational modeling. Experimental research was conducted to investigate the use of the CHR language for this purpose.

The remainder of this paper is organized as follows. Section II describes works related to the two areas that motivated this work: (1) the field of educational computational modeling and (2) the field of abduction. Section III describes ways that the CHR language can be used for computational modeling. The materials and methods, experimental setup, and subjects of the research are presented in sections IV, V, and VI, respectively. Finally, sections VII and VIII present the results, discussion, conclusions, and future work.

## II. RELATED WORK

This work is related to two research fields: (1) computational modeling for educational purposes and (2) abduction.

The first field originated from Papert's studies regarding the LOGO language [17] in the late 1960s. Decades of research on the pedagogical and epistemological aspects [3] of the praxis of Modeling-based Learning (MbL) [4] followed Papert's pioneering studies on computational modeling. In parallel, in the computational field, several general purpose languages, predominantly imperative, object-oriented, and functional (e.g., C, Java, Lisp), were used for modeling.

Specific purpose systems have also been developed, most notably those that have been dedicated to the modeling of dynamic systems, i.e., systems that depend on time. Structural Thinking Experimental Learning Laboratory with Animation (STELLA) [18] and Interacting Quantities Omitting Numbers (IQON) [15] are two examples of these systems. The first is dedicated to quantitative modeling, and the second is used for the semi-quantitative modeling of dynamic systems. The models in STELLA and IQON are graphs in which the dynamic variables are represented as vertices, and the edges determine the relation between them. Mathematically, a STELLA or IQON model represents systems of differential equations. The dynamic variables in a STELLA model assume values of the set of real numbers, whereas the variables of an IQON model assume values on a scale that can vary between none, low, moderate, and high.

Examples of recent studies on educational computational modeling include research on the use of systems such as

978-1-5090-5920-1/17/$31.00 ©2017 IEEE

Scratch [13], MatLab [5], and CellCollective [9]. Scratch has been used primarily for modeling animations, MatLab for mathematical modeling in engineering, physics, economy, and demography, and CellCollective for modeling biological processes.

Abduction is a type of reasoning that aims to formulate hypotheses to explain observed facts, considering a theory as the basis [10]. Psychologically, abduction is applied when, for example, a person creates hypotheses to explain the fact that the engine of an automobile does not start or to diagnose a disease using symptoms presented by a patient as facts. Computational systems for abduction have been proposed as an extension of traditional logic programming via constraint logic programming systems [8]. Examples of these systems include Abductive Logic Programming (ALP) [11], Abductive Constraint Logic Programming (ACLP) [12], Peirce Online [19], Abd1 [16], and CHR [1] [6]. CHR is strongly related to this work; therefore, it is the topic of section III.

## III. Employing the CHR Language for Abductive Computational Modeling

The CHR language was presented as a prototype in [6]. Currently, there are several implementations of this language. In most implementations CHR is not offered as an isolated language but rather as an extension that is combined with the syntax of a host language, such as Prolog, Lisp, Haskel, C, or Java [7]. This work employed an implementation of CHR that is an extension of the Prolog language, which is available in the SWI-Prolog environment [20]. As a linguistic extension of Prolog, it inherits the basic nomenclature of the Prolog language. Thus, the same notions and definitions of Prolog (cf. [2]) apply to the CHR language: constant, variable, atom, term, predicate, arity, rule, clause, head and body of clause, question, goal, use of capital letters for variables, etc.

A CHR program consists of three rule types, whose formats are:

- Simplification: $h_1 , h_2 , ... , h_n$ <=> *Guard* | $b_1 , b_2 , ... , b_m$.
- Propagation: $h_1 , h_2 , ... , h_n$ ==> *Guard* | $b_1 , b_2 , ... , b_m$.
- "Simpagation": $h_1 , h_2 , ... , h_n \setminus h_{n+1} , h_{n+2} , ... , h_p$ <=> *Guard* | $b_1 , b_2 , ... , b_m$.

Each $h_i$ and each $b_i$ are predicates ($n \geq 1$, $m \geq 1$ e $p \geq 2$). The rules are composed of head ($h_i$) and body ($b_i$), in the same way as the Prolog rules. The "," (comma) is the sequence operator and represents a conjunction. In the body of a rule, it is possible to use the operator ";" instead of the operator "," to represent disjunction between predicates. The *Guard* is a set, possibly empty, of predicates separated by the "," operator. An empty *Guard* is interpreted as *true*.

The execution of a CHR program begins with a question (set of predicates). A rule is applied if (1) the rule head coincides with the predicates and (2) the rule guard is satisfied. Simplification rules change a set of predicates from the head of the rule to the set of predicates in the body of the rule. Propagation rules add body predicates without removing the head predicates. For "simpagation" rules, the predicates before the backslash remain, and those after the backslash are removed. CHR rules can be understood as rules used to rewrite predicates, and program execution occurs by match predicates and rules until no match can be made. The unification of atoms occurs the same as in Prolog. A formal description of the operational semantics of CHR is presented in [22].

Abductive reasoning models can be developed and executed using the CHR language available in the SWI-Prolog environment. The abductive theory is described via CHR rules in a CHR program. Facts are presented as questions at the execution prompt ("?") of the SWI-Prolog environment.

**Example 1** (*An abductive model written in CHR language*). In this example, a small model of the starting system of an automotive combustion engine is programmed in the CHR language. The execution of this model computes abductive reasoning on the combustion engine starting system. The example covers the operation and defects of some components that are usually present in most automotive combustion engine starter systems.

The program (Fig. 1) uses predicates with an arity equal to zero, which name mnemonic propositions for "the combustion engine starts, or the combustion engine does not start", "the battery is charged, or the battery is uncharged", "the starter motor is defective, or the starter motor is okay", "the fuel supply system is defective, or the fuel supply system is okay", "the starter motor makes noise when it is triggered, or the starter motor makes no noise", and "the panel lights are on, or the panel lights are off".

Line 1 specifies to the SWI-Prolog system that the "chr" library will be used, and line 2 specifies each predicate used. CHR is a logical and declarative language. The sentences presented in the lines 7 to 9 declare logical relations to the defects or malfunctions that lead to the combustion engine failing to start. Using simplification rules, the program indicates that if the combustion engine does not start, then the battery is uncharged, or the starter motor is defective, or the fuel supply system is defective (line 7). If the starter motor does not make noise when it is turned on, then the battery is uncharged, or the starter itself is defective (line 8). If the panel lights are off, then the battery is uncharged (line 9).

The sentences presented in the lines 10 to 12 declare logical relations for the normal operation of the system. The simplification rules indicate that if the combustion engine starts, then the battery is charged, the starter motor is okay, and the fuel supply system is okay (line 10). If the starter motor makes noise when it is turned on, then the battery is charged, and the starter motor itself is okay. If the panel lights are on, then the battery is charged (line 12).

Sentences 5 and 6 describe, via simplification rules, anonymous logical relations between predicates: uncharged battery is the antonym of battery with charge (line 5), and combustion motor does not start is the antonym of combustion motor starts (line 6). Other antonyms are defined by the program; however, they are not shown in Fig. 1 to enhance the readability of the figure.

The sentences of lines 3 and 4 eliminate duplicates of the same predicate. Thus, a predicate does not appear more than once, unnecessarily, in the same calculated answer. The occurrence of duplicate predicates is a common effect due to the way CHR calculates the answer to a question; however, it is not always desirable, as is the case in this example. Thus, "simpagation" rules, such as the rule for line 3, specify that in the occurrence of two *unchargedBattery* predicates, the first remains, and the second is removed. The program contains sentences for the elimination of duplicates of each of the declared predicates; however, these sentences are not shown in Fig. 1 to enhance readability.

Fig. 2 illustrates two abductive reasoning computations with the program of the Fig. 1 running in the SWI-Prolog environment. These computations produce hypotheses for the combustion engine failing to start. For Fig. 2(a), shown in lines 1 to 8, the engine does not start (*engineDoesNotStart*) and the panel lights are on (*panelLightsOn*) are the facts presented for the abductive reasoning computation. The computation of the reasoning by the system returns as an answer in English that: "The program calculates two hypotheses for the reasoning: (1) the battery is charged, and the starter motor is defective (lines 2 and 3); (2) the battery is charged, and the fuel supply system is defective (lines 5 and 6)". The reasoning of Fig. 2(b), shown in lines 9 to 14, returns the following answer for the facts combustion engine does not start (*engineDoesNotStart*) and starter makes noise when it is triggered (*starterMotorMakesNoise*): "The program calculates only one hypothesis for the reasoning: the battery is charged, the starter motor is okay, and the fuel supply system is defective".

Models can be continually enhanced to incorporate new elements. For example, the model of Fig. 1 could be improved to provide details of the parts of the fuel supply system. When the student completes this step, the student's knowledge regarding the system modeled improves. That is, the student learns via the modeling process.

```
1.  :- use_module(library(chr)).
2.  :- chr_constraint   unchargedBattery, chargedBattery,
                        engineDoesNotStart, engineStarts,
                        ...  .
3.  unchargedBattery \ unchargedBattery <=> true.
4.  chargedBattery \ chargedBattery <=> true.
    ...
5.  unchargedBattery , chargedBattery <=> fail.
6.  engineDoesNotStart, engineStarts <=> fail.
    ...
7.  engineDoesNotStart <=> unchargedBattery ;
      starterMotorDefect ; fuelSystemDefect.
8.  starterMotorDoesNotMakeNoise <=> unchargedBattery ;
      starterMotorDefect.
9.  panelLightsOff <=> unchargedBattery.

10. engineStarts <=> chargedBattery ,
      starterMotorOK , fuelSystemOK.
11. starterMotorMakesNoise <=> chargedBattery ,
      starterMotorOK.
12. panelLightsOn <=> chargedBattery.
```

Fig. 1. A model of the starting system of an automotive combustion engine programmed in the CHR language.

```
1.  ?- engineDoesNotStart , panelLightsOn.
2.  chargedBattery
3.  starterMotorDefect
4.  true ;
5.  chargedBattery
6.  fuelSystemDefect
7.  true ;
8.  false.
                    (a)
9.  ?- engineDoesNotStart , starterMotorMakesNoise.
10. chargedBattery
11. starterMotorOK
12. fuelSystemDefect
13. true ;
14. false.
                    (b)
```

Fig. 2. Two abductive reasoning computations with the program of the Fig. 1 running in the SWI-Prolog environment**.**

## IV. MATERIALS AND METHODS

The modeling process comprises a cycle in which the student (1) writes (program) the model, (2) executes the model considering a set of facts, and (3) evaluates the execution result. Three skills are required for a student to participate of the modeling cycle: (1) ability to write (program) the model, (2) ability to propose facts to execute the model, and (3) ability to evaluate the results of the execution of a model. This experimental research investigated these three abilities.

A Form was developed for all phases of the experimental research sessions. It was previously validated in a pilot experiment involving two people that small problems were removed, and it was improved. Appendix I describes the Form, and this section discusses its content.

The Form consisted of four parts, each designed to assist a specific phase of the experimental procedure, which will be described in the next section. The Form did not request the personal identification of the participant; only a numeric code was assigned to the parts of the Form so that the answers given by the participants could be grouped at the end of the experiment.

Part 1 aimed to determine the profile of the participant: gender, age, course, and class of the participant.

Part 2 contained a single exercise (Q1) that required the ability to develop a model about the headlights system of a Jeep using the CHR language. The components and logical relations about the Jeep's headlights system were indicated in the exercise. As will be discussed in the next section, this exercise was preceded by an exposition on the CHR language, the SWI-Prolog environment, and the functioning of a Jeep's headlights system. The participants were required to program the model using the CHR language, to compile it, run it, and modify it as many times as desired. After the tasks were completed, the participants were expected to produce a file with the developed program. To facilitate the work, the participants received the file "MyModel.pl" stored in the computer desktop, with the header of the CHR program previously filled in (sections ": - use_module (library (chr))", and ": - chr_constraint").

Part 3 contained exercises (Q2 to Q5) that required the ability to propose facts for performing abductive reasoning using a ready-made model ("JeepHeadlights.pl") of the Jeep's headlights system.

Part 4 of the Form contained exercises that required the ability to evaluate the results of model execution, i.e., the ability to read, interpret, and judge the results of model execution. Two sets of exercises were proposed. The first set of exercises (Q6 to Q9) required the evaluation of the results of the execution of the "JeepHeadlights.pl" model for the facts answered in exercises Q2, Q3, Q4, and Q5. This set of exercises was carefully planned so that situations would arise in which the execution of the model:

- Generated hypotheses that explained the facts presented.
- Did not generate hypotheses for the facts presented.
- Generated only non-explanatory hypotheses, i.e., the resulting hypotheses were the facts presented.

In the second set of exercises (Q10 to Q14), results of abductive reasoning performed in the SWI-Prolog system were presented. In each exercise, the participants were asked to indicate alternatives that they considered correct among several available alternatives.

## V. Experimental Setup

The research encompassed experimental sessions with the participation of a maximum of five subjects in five workstations of a laboratory. Each workstation was equipped with a desk, a chair, a computer with an Intel i3 processor, 3.3 GHz, 4 GB of RAM, running Windows 10 Pro, a text editor, and a compiler for the CHR language available in the SWI-Prolog environment. Each experimental session was composed of four phases.

Phase 1, scheduled to last five minutes, began with the reception, greeting, and thanks to the participants by voluntary and gratuitous contribution. With the help of a multimedia projector, the researcher explained the purpose of the experiment, distributed Part 1 of the Form, and asked participants to answer questions related to their profiles. After the participants answered the questions of Part 1 of the Form, the researcher collected the answers.

Phase 2, which included an exercise (Q1) that required the ability to write (to program) a model using the CHR language and was scheduled to last 60 minutes, began with the distribution of Part 2 of the Form to the participants. Using a multimedia projector, the researcher presented photos of a Jeep and explained the functioning of the Jeep's headlights system and the starting system of the Jeep's combustion engine. The researcher taught the fundamentals of programming in the CHR language using the model in Fig. 1 (starting system of the Jeep's combustion engine) and explained how to compile and execute a CHR program in the SWI-Prolog environment. The pedagogical strategy was learned through examples, without theoretical formalizations. The researcher then requested that the participants create the model requested and that they did so using the logical relations presented in the exercise Q1. During the development of the model, the researcher acted as a mediator of a modeling environment, guiding participants and discussing problems without presenting solutions. Participants were required to continue the cycle of writing the model, executing it, and evaluating results until they considered the work to be finished or until the end of the time determined for the phase, which was announced by the researcher. At the end of the phase, the researcher collected the files containing the models developed by the participants using a pen drive.

Phase 3, which included exercises that required the ability to propose facts for abductive reasoning and was scheduled to last 10 minutes, began with the distribution of Part 3 of the Form to the participants. The researcher offered each participant a file containing a ready-made model, "JeepHeadlights.pl," for the Jeep's headlights system and requested that they follow the instructions on the Form. After completing the exercises, the researcher collected the answers.

Phase 4, which included exercises requiring the ability to evaluate the results of the execution of model and was scheduled to last 15 minutes, began with the distribution of Part 4 of the Form. The first set of exercises was performed by running the "JeepHeadlights.pl" model and interpreting the results of the execution. The second set of exercises required selecting an alternative among the multiple available alternatives for each exercise. After completing the exercises, the researcher collected the answers.

The experimental session ended with a five-minute debriefing in which the researcher made general comments regarding the solution of the exercises, allowed participants to comment on their solutions, and thanked the participants.

## VI. Subjects

A total of 88 students in a senior high school course offered by the Federal Institute of Education, Science and Technology of South Minas Gerais, Muzambinho, Brazil, were randomly selected to participate in the experiment. The research population is composed of 160 students. The participants were young people seeking technical training in Informatics simultaneously with a classical secondary education. Approximately half were female (47%), and half were male (53%), ranging in age from 16 to 19 years with mean and median ages equal to 17 years. The minimum sample size, 75 students, was calculated assuming the following parameters: population size equal to 160, significance level equal to 5%, confidence level equal to 95%, and heterogeneity-homogeneity of the population equal to 90%.

## VII. Results and Discussion

For the quantitative analysis of the results, the Pattern of Answers described in Appendix II was used as a "template". A numerical scale from 0 to 10 was used to measure performance in phases 2, 3, and 4. Participants' answers were analyzed according to the following criteria and metrics:

- *Part 2 – Syntactic and semantic correctness of the model developed:* The model is compiled, and the

existence or absence of syntax errors is evaluated. If there is syntax error, the value 0 (zero) is assigned. If there is no syntax error, the result of the execution of the model developed is compared with the result of the execution of the model of the Pattern of Answers, described in Part 2 of Appendix II, for each of the ten reasoning computations activated by the facts:

1. *leftHeadlightOn.*
2. *leftHeadlightOff.*
3. *rightHeadlightOn.*
4. *rightHeadlightOff.*
5. *leftHeadlightOn* , *rightHeadlightOn*.
6. *leftHeadlightOn* , *rightHeadlightOff.*
7. *leftHeadlightOff* , *rightHeadlightOn.*
8. *leftHeadlightOff* , *rightHeadlightOff.*
9. *unchargedBattery*.
10. *leftHeadlightOn* , *rightHeadlightOn* , *unchargedBattery*.

Add 1 for each result that matches the result computed by the model of the Pattern of Answers.

- *Part 3 – Correctness of the proposed facts*: The answer is compared with the answer assumed to be correct in the Pattern of Answers (Appendix II). For each of the four proposed exercises (Q2, Q3, Q4, and Q5), the value 1 is assigned when there is a coincidence between the answers; the value 0 is assigned otherwise. The values assigned to the exercises are summed. The sum is multiplied by 10/4 to obtain the normalized score for the scale in the interval [0..10].
- *Part 4 – Correctness of the evaluation of the results*: The answer is compared with the answer assumed to be correct in the Pattern of Answers (Appendix II). For each of the nine proposed exercises (Q6 to Q14), the value 1 is assigned when there is a coincidence between the answers; the value 0 is assigned otherwise. Add the values assigned to the exercises and multiply this sum by 10/9 to obtain the normalized exercise score for the scale in the interval [0..10].

Table I presents the mean and standard deviation of the scores obtained by the participants in phases 2, 3, and 4 of the experiment.

In general, these results suggest a good performance of participants in the proposed exercises. This is also true for phase 4, which included exercises requiring the ability to evaluate the results of model execution, for which the mean was lower than in the others.

The following question was also investigated: is the variation of the means a causality? To examine this question, a statistical analysis was conducted using the following hypotheses:

- $H_0$: There is no difference between the means obtained in phases 2, 3, and 4 of the experiment.
- $H_A$: There is a difference between the means obtained in phases 2, 3, and 4 of the experiment.

TABLE I. TABLE I. RESULTS (MEAN AND STANDARD DEVIATION) OF THE SCORES OBTAINED BY PARTICIPANTS IN PHASES 2, 3, AND 4.

| Phase | *n* | Mean (0 – 10) ± SD |
|---|---|---|
| 2 | 88 | 8.27 ± 2.66 |
| 3 | 88 | 9.35 ± 1.56 |
| 4 | 88 | 7.46 ± 1.90 |

An analysis of variance (ANOVA) [14] was carried out with a posteriori treatment by Student's *t*-tests for paired data with a significance level equal to 0.05 (5%). For the population studied, the ANOVA test suggests that there is a significant difference between the means ($F(2, 261) = 17.92$, $p < 0.0001$), that is, the null hypothesis $H_0$ should be rejected, and the alternative hypothesis $H_A$ should be accepted. Student's *t*-tests suggest that there is a significant difference between means compared two by two:

- Phase 2 and Phase 3 ($t = 3.40$, $p < 0.0001$).
- Phase 2 and Phase 4 ($t = 2.57$, $p = 0.0022$).
- Phase 3 and Phase 4 ($t = 5.96$, $p < 0.0001$).

That is, although the performance of the studied population was good in all phases, there is a statistically significant difference between the performance in each of the phases. In descending order of the mean is the performance regarding the ability to: (1) propose facts (Phase 3); (2) program a model using the CHR language (Phase 2), and (3) evaluate the results of model execution (Phase 4).

Among the three abilities investigated, it is not surprising that the proposition of facts (Phase 3) had the best performance because it simply requires the selection of terms that will be used as facts.

It can be conjectured that the large difference between the natural language and the language used by the SWI-Prolog environment to present results negatively affected performance in evaluating the results of model execution (Phase 4). For example, the sentence presented by the SWI-Prolog environment, shown in Fig. 2(a), is quite different from a sentence that could be written in English, such as: "Two hypotheses have been computed: (1) the battery is charged, and the starter motor is defective; (2) the battery is charged, and the fuel supply system is defective".

On the other hand, it is surprisingly positive to note that the population studied successfully overcame the challenges of writing sentences in the CHR language, which was an ability required in Phase 2.

Qualitatively, how can these results be understood? Based on the interactions with the participants during the experimental sessions, the researchers subjectively believe that the participants' performance was good. Although this evaluation coincides with the quantitative evaluation, it is necessary to observe that the models used in the experiment

were limited to Propositional Logic (predicates with arity equal to zero) for simplicity. These models are suitable for beginners with no experience in the use of the CHR language, as was the case for all participants in the experimental research. More complex models, involving predicates with an arity greater than zero or a logical circularity between predicates, for example, were not utilized in the research. This does not diminish the value of the research carried out but rather circumscribes its scope, and it defines the value that is appropriate for the results found.

## VIII. Conclusions and Future Work

On the one hand, educational computational modeling is a relevant activity for research and is widely valued in educational practice. On the other hand, computational systems for abduction have been developed and studied in several non-educational contexts. This work proposes and experimentally investigates the use of the CHR language to create an educational environment for abductive computational modeling.

For the population studied, the results suggest the feasibility of using the CHR language in the context of educational computational modeling. It is not possible to guarantee that these results can be reproduced by other populations with different levels of knowledge and interest, but they do create a positive expectation due to the good performance of the studied population in the proposed activities.

The results indicate important technical and pedagogical challenges that must be overcome as well. The evaluation of the results of the execution of a model is an essential activity in a modeling environment. Difficulties in the evaluation process create barriers to the continuity of the modeling cycle, which negatively affects the learning process via modeling activities. This is an important drawback that should be addressed with measures of a technical or pedagogical nature. Future works of a pedagogical nature could contribute to alleviating the difficulties imposed by the presentation language of the results of the SWI-Prolog system. Works of a technical nature could contribute to developing a presentation language that is simpler to learn and use.

## Acknowledgments

We would like to thank all subjects who volunteered for the experiment for their invaluable contribution to this work.

## Appendix I – Form Used in Experimental Research (condensed format for publication)

**Part 1**: Determining the profile of the participants (gender, age, course, class).

**Part 2**: Ability to program a model using the CHR language.

Three pictures of a Jeep are presented in a multimedia projector. The first one depicts the Jeep by a front and side angle. The second one depicts the instrument panel with textual highlights for the button that activates the headlights and for the ignition switch. The third depicts the engine compartment with textual highlights for the battery and starter motor.

**Q1)** Using a computer, develop a model for the Jeep's headlights system in the CHR language. You must use the terms and relations expressed below:

**leftHeadlightOff**: *unchargedBattery* **or** *headlightSwitchedToTheOff* **or** *burnedOutLeftHeadlight*.

**rightHeadlightOff**: *unchargedBattery* **or** *headlightSwitchedToTheOff* **or** *burnedOutRightHeadlight*.

**leftHeadlightOn**: *chargedBattery* **and** *headlightSwitchedToTheOn* **and** *leftHeadLightOK*.

**rightHeadlightOn**: *chargedBattery* **and** *headlightSwitchedToTheOn* **and** *rightHeadLightOK*.

To make your work easier, use the "MyModel.pl" file saved in the computer desktop. The header that is necessary for you to develop your model is already edited in the file.

**Part 3**: Ability to propose facts

You have received a file with a ready-made model called "JeepHeadlights.pl," which explains why the Jeep headlights are off or on. With the template open in an editor, please answer the following exercises.

**Q2)** What facts should you use to compute hypotheses to explain why the Jeep's left headlight is off?

**Q3)** What facts should you use to compute hypotheses to explain why the Jeep's left headlight is on and the Jeep's right headlight is off?

**Q4)** What facts should you use to compute hypotheses to explain why the headlights are on and the battery is uncharged?

**Q5)** What facts should you use to compute hypotheses to explain why the battery is uncharged?

**Part 4**: Ability to evaluate the results of the execution of a model.

***First set of exercises***

In the following exercises, write in English your interpretation about the results of execution of the "JeepHeadlights.pl" model:

**Q6)** Using the facts that you answered in exercise Q2.

**Q7)** Using the facts you answered in exercise Q3.

**Q8)** Using the facts that you answered in exercise Q4.

**Q9)** Using the facts you answered in exercise Q5.

***Second set of exercises***

Using Table II, mark one, and only one, correct alternative in each of the following exercises.

**Q10)** In relation to the reasoning described in column Q10 of Table II, what can be stated?

**(a)** The fact is false. **(b)** The model has no hypotheses that explains the fact. **(c)** The fact does not belong to the model. **(d)** The fact, associated with another fact, is true.

**Q11)** In relation to the reasoning described in column Q11 of Table II, what can be stated?

**(a)** Fact1 is true, and fact2 is false. **(b)** Fact2 is true, and fact1 is false. **(c)** The hypotheses computed are the facts presented. **(d)** Fact1 or fact2 are possible explanations.

**Q12)** In the well-known Little Red Riding Hood story, consider that Little Red Riding Hood has received, as Grandma's answers to her questions, what is described in column Q12 of Table II. This means that:

**(a)** The hypothesis computed by the Little Red Riding Hood character is consistent with the story, considering the fantasy model of the story. **(b)** The hypothesis computed by the Little Red Riding Hood character is consistent with a model whose intention is to imitate reality. **(c)** The hypothesis computed by the Little Red Riding Hood character is partially coherent with the story, considering the fantasy model of the story. **(d)** The Little Red Riding Hood character does not compute a hypothesis about the facts presented.

**Q13)** In relation to the reasoning described in column Q13 of Table II, what can be stated?

**(a)** The execution did not generate hypotheses. **(b)** Execution generated two hypotheses meaning "a or b". **(c)** The execution generated two hypotheses meaning "a and b". **(d)** The execution generated a true and a false hypothesis.

**Q14)** In relation to the reasoning described in column Q14 of Table II, what can be stated?

**(a)** The execution did not generate hypotheses. **(b)** The execution generated two hypotheses meaning "(a and b) or c". **(c)** The execution generated a hypothesis meaning "a and b and c". **(d)** The execution generated a hypothesis signifying "a or b or c".

TABLE II. SOME RESULTS OF ABDUCTIVE REASONING.

| Q10 | Q11 | Q12 | Q13 | Q14 |
|---|---|---|---|---|
| ? – fact.<br><br>false. | ? – fact1 , fact2.<br><br>fact1<br>fact2<br>true ;<br>false. | ? – toSeeYouBetter , toSmellYouBetter , toHearYouBetter , toEatYou.<br><br>wolf<br>true ;<br>false. | ? – fact.<br><br>a<br>b<br>true ;<br>false. | ? – fact.<br><br>a<br>b<br>true ;<br>c<br>true ;<br>false. |

## APPENDIX II - PATTERN OF ANSWERS TO THE EXERCISES

### Part 2

**Q1**:

:- *use_module*(*library*(*chr*)).

:- *chr_constraint rightHeadlightOff, rightHeadlightOn*,

…

*leftHeadlightOn* \ *leftHeadlightOn* <=> true.

*unchargedBattery* \ *unchargedBattery* <=> true.

…

*rightHeadlightOff* , *rightHeadlightOn* <=> fail.

*unchargedBattery* , *chargedBattery* <=> fail.

…

*leftHeadlightOff* <=> *unchargedBattery* ; *headlightSwitchedToTheOff* ; *burnedOutLeftHeadlight*.

*rightHeadlightOff* <=> *unchargedBattery* ; *headlightSwitchedToTheOff* ; *burnedOutRightHeadlight*.

*rightHeadlightOn* <=> *chargedBattery* , *headlightSwitchedToTheOn* , *rightHeadLightOK*.

*leftHeadlightOn* <=> *chargedBattery* , *headlightSwitchedToTheOn* , *leftHeadLightOK*.

### Part 3

**Q2**: *leftHeadlightOff*.

**Q3**: *leftHeadlightOn* , *rightHeadlightOff*.

**Q4**: *leftHeadlightOn* , *rightHeadLightOK* , *unchargedBattery*.

**Q5**: *unchargedBattery*.

### Part 4

**Q6**: The battery is uncharged, or the headlight switch is off, or the left headlight is burned out.

**Q7**: The right headlight is burned out, the left headlight is okay, the headlight switch is on, and the battery is charged.

**Q8**: There are no hypotheses to explain the facts.

**Q9**: The hypothesis is that the battery is uncharged (hypothesis does not explain anything).

**Q10**: b.

**Q11**: c.

**Q12**: a.

**Q13**: c.

**Q14**: b.

## REFERENCES


[1] S. Abdennadher and H. Christiansen, "An experimental CLP platform for integrity constraints and abduction," in Proceedings of the Flexible Query Answering Systems: Advances in Soft Computing Series (FQAS), 2000, pp. 141–152.

[2] M. Bramer, Logic programming with prolog, 2nd ed., London: Springer, 2013.

[3] T. Campbell and P. Oh, "Engaging students in modeling as an epistemic practice of science," Journal of Science Education and Technology, vol. 24, n. 2, 2015, pp. 125–131.

[4] J. Clement and M. A. Rea-Ramirez, Model based learning and instruction in science: models and modeling in science education, J. Clement and M. A. Rea-Ramirez, Eds. Dordrecht: Springer, 2007.

[5] A. B. Downey, Physical modeling in MATLAB. Needham: Green Tea Press, 2011.

[6] T. Frühwirth, "Theory and practice of constraint handling rules," The Journal of Logic Programming, vol. 37, n. 3, 1998, pp. 95–138.

[7] T. Frühwirth, Constraint handling rules. Cambridge: Cambridge University Press, 2009.

[8] J. Jaffar, M.J. Maher, "Constraint logic programming: a survey," Journal of Logic Program, vol. 19, n. 20, 1994, pp. 503–581.

[9] T. Helikar et al., "Integrating interactive computational modeling in biology curricula," PLOS computational biology vol. 11, n. 3, e1004131, 2015.

[10] J. R. Josephson and S. G. Josephson, Abductive inference: computation, philosophy, technology. Cambridge: Cambridge University Press, 1994.

[11] A. C. Kakas, R. A. Kowalski, and F. Toni, Abductive logic programming. Journal of Logic and Computation, vol. 2, n. 6, 1992, pp. 719–770.

[12] A. C. Kakas, A. Michael, and C. Mourlas, ACLP: Abductive constraint logic programming. Journal of Logic Programming, vol. 44, 2000, pp. 129–177.

[13] V. Lopez, and M. I. Hernandez, "Scratch as a computational modelling tool for teaching physics," Physics Education, vol. 50, n. 3, 2015, pp. 310–316.

[14] J. T. McClave and T. T. Sincich, Statistics, 12th ed. London: Pearson Education, 2014.

[15] R. J. Miller, J. Ogborn, J. H. Turner, D. R. Briggs, and D. Brough, "Towards a tool to support semi-quantitative modeling," in Proceedings of the International Conference on Advanced Research on Computers in Education, July 1990.

[16] O. L. Oliveira, C. E. A. Oliveira, R. J. Martins, and M. Matsumoto, "A logic programming language designed for the modeling of abductive reasoning in an educational context," in Procedings of the Mexican International Conference on Artificial Intelligence (MICAI), 2016, pp. 1–13.

[17] S. Papert, Mindstorms: children, computers, and powerful ideas. 2nd ed. New York: Basic Books, 1993.

[18] B. Richmond, S. Peterson, and P. Vescuso, An academic user's guide to STELLA. Lyme: High Performance Systems, 1987.

[19] F. Rodrigues, C. E. A. Oliveira, and O. L. Oliveira, "Peirce: an algorithm for abductive reasoning operating with a quaternary reasoning framework," Research in Computer Science, vol. 82, 2014, pp. 53–66.

[20] SWI-Prolog-Org, SWI-Prolog organization, 2017. Available in http://www.swi-prolog.org. Acessed: 22 April 2017.

[21] D. Walton, Abductive reasoning, eBook ed. Alabama: The University of Alabama Press, 2014.

[22] G. J. Duck, P. J. Stuckey, M. J. G. Banda, and C. Holzbaur, "The refined operational semantics of constraint handling rules," in Proceeedings of the 20th International Conference on Logic Programming (ICLP), 2004, pp. 90–104.