Abstract

An AI Approach to Large Scale Medical Appointment (Re)Scheduling using ASP

Kanias Stavros

Bachelor student at the University of Patras

Faculty advisor Prof. Pavlos Peppas

up1066563@upnet.gr

Despite their significant solving capabilities ASP solvers haven’t yet crossed the line between academic and widespread industrial usage. Even in areas where classical AI solutions developed with the ASP framework exceed solutions developed with procedural algorithms in common programming languages in terms of raw performance, the later approach is most commonly chosen. Τhe lack of integration between ASP solvers and integral parts of standard software development components, such as SQL databases, is commonly put forward. In this paper a Python interface for converting a database to a logic program knowledge base is presented. The solution was utilized to solve the problem of medical appointment rescheduling yielding great results.

# 1 Introduction

Since its inception in 1972, logic programming has developed to the point which contemporary solvers, such as clingo, are able to solve very complex (mostly hard NP-complete) problems when they are expressed under answer set semantics. There has been though, a technical gap between the ASP theory and the industrial applications which most of the time involve a large-scale database. To be more precise, as of today a solution to create a knowledge base for an ASP solver from a database with an automated process has not been developed. In this paper the form and use of a simple interface between a PostgreSQL database and a clingo logic program is described. The final software solution greatly simplifies the data flow between the database and the solver bringing the possibility of using ASP solvers in the backend part of industrial scale software applications to the spotlight. The programming language used for its development was Python and the code was based on the Potassco backed CLORM interface module. The problem of appointment scheduling and rescheduling was chosen to showcase the immense power of ASP over classical programming techniques in applications that require the process of large-scale data combinations and classical maximization algorithms. The healthcare framework was also chosen to make the application more practical and also to provide a clear example of the ways society can benefit from the use of an AI approach to (re)scheduling problems. In Section 2 a more in-depth examination of the medical appointment (re)scheduling problem is presented followed by a simple example comparing two approaches to the problem. In Section 3 the ASP solution to the problem is presented. First, the structure of the SQL database is analyzed and then follows the logic program used to solve the problem. The third subsection is dedicated to the form and usage of the Python interface is described. Finally, in Section 4, various statistics describing the performance of the solution are presented extracted from different datasets followed by comments on the performance limitations and possible future optimization.

# 2 The Problem

Assuming a hospital with thousands of patients and dozens of doctors. Each doctor works for 8 hours providing 8 one- hour appointment timeslots. In most contemporary healthcare systems patients can book an appointment only if it is currently available. As soon as a vacant timeslot is requested it is granted by the system which blocks other patients from requesting the same timeslot. The problem that arises from this policy is that in the case of a cancelation the timeslot will again become available but will most likely be wasted since no waiting queue was developed in order for another patient to claim it instantly. Even in the case where a waiting queue exists an ‘individual first’ approach will be followed, and the system will grant the request with the highest score for the timeslot in question from the queue. In this case, given that the patients will freely choose a better timeslot for themselves if asked, the common benefit is sacrificed. As described in the following example, using an AI approach to maximization, if the system chooses not to give the canceled appointment to the second-best request of the queue a chain-reaction of rescheduled appointments can occur through which a much better common benefit can emerge.

A scenario that allows patients to select more than one timeslot with ascending preference order, where 6 patients claim 4 timeslots can be seen in the following table. Just to clarify, the scores are derived from the formula:

|  |  |  |
| --- | --- | --- |
| Timeslot | Patient | Request Score |
| Monday at 9 am | Nikos (Granted) | 90 |
|  | Giorgos  Kostas | 87  86 |
| Tuesday at 1 pm | Giorgos (Granted)  Ioannis  Maria | 72  60  55 |
| Thursday at 11 am  Friday at 3 pm | Ioannis (Granted)  Maria  Kostas (Granted)  Despoina | 45  40  71  70 |
|  |  |  |

Table 1: Initial Schedule

In the above scenario, if we define the common benefit as the sum of the scores of the granted requests, a common benefit of 278 is offered by the healthcare system to society. In the case where Nikos cancels his appointment, the first timeslot becomes available. If an ‘individual first’ approach is used the timeslot will be granted to Giorgos and thus a common benefit of 258 will emerge. In contrast the AI approach mentioned above and described in depth in the following paragraph will result in the following schedule.

|  |  |  |
| --- | --- | --- |
| Timeslot | Patient | Request Score |
| Monday at 9 am | Kostas (Granted) | 86 |
|  |  |  |
| Tuesday at 1 pm | Giorgos (Granted)  Ioannis  Maria | 72  60  55 |
| Thursday at 11 am  Friday at 3 pm | Ioannis (Granted)  Maria  Despoina | 45  40  70 |
|  |  |  |

Table 2: Schedule after AI rescheduling

The result is a schedule that provides a higher common benefit of 273.

# 3 The Solution

## 3.1 Database Design

Print manuscripts two columns to a page, in the manner in which these instructions are printed. The exact dimensions for pages are:

## 3.2 Optimal (Re)Scheduling with ASP

Print manuscripts two columns to a page, in the manner in which these instructions are printed. The exact dimensions for pages are:

* left and right margins: .75*"*
* column width: 3.375*"*
* gap between columns: .25*"*
* top margin—first page: 1.375*"*
* top margin—other pages: .75*"*
* bottom margin: 1.25*"*
* column height—first page: 6.625*"*
* column height—other pages: 9*"*

## 3.3 The Python component

For the production of the electronic manuscript, you must use Adobe’s *Portable Document Format* (PDF). A PDF file can be generated, for instance, on Unix systems using ps2pdf or on Windows systems using Adobe’s Distiller. There is also a website with free software and conversion services: http://www.ps2pdf.com/. For reasons of uniformity, use of Adobe’s *Times Roman* font is strongly suggested. In LATEX2e, this is accomplished by putting

\usepackage{times}

in the preamble.[[1]](#footnote-1)

Additionally, it is of the utmost importance to specify the American **letter** format (corresponding to 8-1/2*"*  11*"*) when formatting the paper. When working with dvips, for instance, one should specify –t letter.

# 4 Evaluation

Place the abstract at the beginning of the first column 3*''* from the top of the page, unless that does not leave enough room for the title and author information. Use a slightly smaller width than in the body of the paper. Head the abstract with “Abstract” centered above the body of the abstract in a 12-point bold font. The body of the abstract should be typeset in a 9-point font.

The abstract should be a concise, one-paragraph summary describing the general thesis and conclusion of your paper. A reader should be able to learn the purpose of the paper and the reason for its importance from the abstract. The abstract should be no more than 200 words long.

## 2.5 Text

The main body of the text immediately follows the abstract. Use 10-point type in a clear, readable font with 1‑point leading (10 on 11).

Indent when starting a new paragraph, except after major headings.

## 2.6 Headings and Sections

When necessary, headings should be used to separate major sections of your paper. (These instructions use many headings to demonstrate their appearance; your paper should have fewer headings). All headings should be capitalized using Title Case.

### Section Headings Print section headings in 12-point bold type in the style shown in these instructions. Leave a blank space of approximately 10 points above and 4 points below section headings. Number sections with arabic numerals.

### Subsection Headings Print subsection headings in 11-point bold type. Leave a blank space of approximately 8 points above and 3 points below subsection headings. Number subsections with the section number and the subsection number (in arabic numerals) separated by a period.

### Subsubsection Headings Print subsubsection headings in 10-point bold type. Leave a blank space of approximately 6 points above subsubsection headings. Text follows the subsection heading in the same line after a double space, like in this paragraph itself.

### Acknowledgements You may include an unnumbered acknowledgements section, including acknowledgments of help from colleagues, financial support, and permission to publish. If present, acknowledgements must be in a dedicated, unnumbered section appearing after all regular sections but before any appendices or references.

Use

\section\*{Acknowledgements}

to typeset the acknowledgements section in LATEX.

### Appendices Any appendices directly follow the main text and look like sections, except that they are numbered with capital letters instead of arabic numerals. See this document for an example.

### Supplementary Material Authors may optionally provide supplementary material (e.g. proof details, additional experimental results) as a separate file. Such material will be consulted at the discretion of reviewers and will not be published. Please refer to the conference website for further information.

### References The references section is headed “References”, printed in the same style as a section heading without numeration. A sample list of references is given at the end of these instructions. Use a consistent format for references. The reference list should not include unpublished work.

## 2.7 Citations

Citations within the text should include the author’s last name and the year of publication, for example (Gottlob, 1992). Append lowercase letters to the year in cases of ambiguity. Treat multiple authors as in the following examples: (Abelson *et al.*, 1985) or (Baumgartner *et al.*, 2001) (for more than two authors) and (Brachman and Schmolze, 1985) (for two authors). If the author portion of a citation is

obvious, omit it, e.g., Nebel (2000). Collapse multiple citations as follows: (Gottlob *et al.*, 2002; Levesque, 1984a).

## 2.8 Footnotes

Place footnotes at the bottom of the page in a 9-point font. Refer to them with superscript numbers.[[2]](#footnote-2) Separate them from the text by a short line.[[3]](#footnote-3) Avoid footnotes as much as possible; they interrupt the flow of the text.

# 3 Illustrations

Place all illustrations (figures, drawings, tables, and photographs) throughout the paper at the places where they are first discussed, rather than at the end of the paper.

They should be floated to the top (preferred) or bottom of the page, unless they are an integral part of your narrative flow. When placed at the bottom or top of a page, illustrations may run across both columns, but not when they appear inline.

|  |  |  |
| --- | --- | --- |
| Scenario | δ(s) | Runtime(ms) |
| Paris | 0.1 | 13.65 |
|  | 0.2 | 0.01 |
| New York | 0.1 | 92.50 |
| Singapore | 0.1 | 33.33 |
|  | 0.2 | 23.01 |

Table 1: Example table

**Algorithm 1** Example Algorithm

**Input**: Your algorithm’s input

**Parameter**: Optional list of parameters

**Output**: Your algorithm’s input

1: Let t= 0.

2: **while** condition **do**

3: Do some action.

4: **if** conditional **then**

5: Perform task A.

6: **else**

7: Perform task B.

8: end if

9: end while

10: **return** solution

Illustrations must be rendered electronically or scanned and placed directly in your document. All illustrations should be understandable when printed in black and white, albeit you can use colors to enhance them. Line weights should be 1/2-point or thicker. Avoid screens and superimposing type on patterns, as these effects may not reproduce well.

Number illustrations sequentially. Use references of the following form: Figure 1, Table 2, etc. Place illustration numbers and captions under illustrations. Leave a margin of 1/4-inch around the area covered by the illustration and caption. Use 9-point type for captions, labels, and other text in illustrations. Captions should always appear below the illustration.

# 4 Tables

Tables are considered illustrations containing data. Therefore, they should also appear floated to the top (preferably) or bottom of the page, and with the captions below them. If you are using MS Word, we recommend inserting tables inside a text box to ease positioning.

It is encouraged to use the format shown in Table 1, where there are no vertical lines and only three horizontal ones: Two thick lines (1 ½ points) on top and bottom of the table and one thin line (1 point) below the header.

Notice also that numeric columns are right aligned, making it easier to compare the numbers. Make sure to also right-align the corresponding headers, and to use the same precision for all numbers. Also, try to avoid unnecessary repetition, both between lines (no need to repeat the scenario name in this case) as well as in the content (units can be shown in the column header).

# 5 Formulas

KR's two-column format makes it difficult to typeset long formulas. If your paper contains a significant number of equations, we strongly recommend using the LATEX document preparation system.

In your document, equation numbers should be typeset with the same font and size than the main text (10pt). Main symbols of the formula should not be smaller than 9pt.

*f(x)=ax+b* (1)

# 6 Examples, Definitions, Theorems and Similar

Examples, definitions, theorems, corollaries and similar must be written in their own paragraph. The paragraph must be separated by at least 2pt and no more than 5pt from the preceding and succeeding paragraphs. They must begin with the kind of item written in 10pt bold font followed by their number (e.g.: Theorem 1), optionally followed by a title/summary between parentheses in non-bold font and ended with a period. After that the main body of the item follows, written in 10pt italics font (see below for examples).

Example 1 (How to write an example). This is a titled example.

Theorem 1 This is an example of an untitled theorem.

# 7 Proofs

Proofs must be written in their own paragraph separated by at least 2pt and no more than 5pt from the preceding and succeeding paragraphs. Proof paragraphs should start with the keyword ``Proof." in 10pt italics font. After that the proof follows in regular 10pt font. At the end of the proof, an unfilled square symbol (qed) marks the end of the proof.

Proof. This paragraph is an example of how a proof should look like. □

# 8 Algorithms and Listings

Algorithms and listings are a special kind of figures. Like all illustrations, they should appear floated to the top (preferably) or bottom of the page. However, their caption should appear in the header, left-justified and enclosed between horizontal lines, as shown in Algorithm 1. The algorithm body should be terminated with another horizontal line. It is up to the authors to decide whether to show line numbers or not, how to format comments, etc.

We suggest placing the algorithm inside a text box to ease positioning when using MS Word.

9 LATEX and Word Style Files

The LATEX style file is kr.sty and the BibTEX style file to use is kr.bst. and kr.bib. The LATEX file kr-instructions.tex, containing the source of the present document, and the BibTEX file kr-sample.bib, containing some example BibTEX entries, may serve as a formatting sample (these two files are not needed for typesetting your paper). The LATEX style file is for version 2e of LATEX, and the BibTEX style file is for version 0.99c (*not* version 0.98i) of BibTEX. Note that the kr.sty style differs from the kr.sty file used for KR2020 but remains unchanged since then.

The Microsoft Word style file consists of a single file, kr20.docx, which may serve as a formatting sample for Microsoft Word users. Please make use of the ad-hoc styles that have been defined for the different parts of the document, and that are listed in the Styles Pane. Note that this template differs from the one used for KR2020 but remains essentially unchanged since then.

Further information on using these styles for the preparation of papers for KR2023 can be obtained by contacting kr.proceedings@confdna.com.

# Acknowledgments

The preparation of these instructions and the LATEX and BibTEX files that implement them was supported by Schlumberger Palo Alto Research, AT&T Bell Laboratories, and Morgan Kaufmann Publishers. Preparation of the Microsoft Word file was supported by IJCAI. An early version of this document was created by Shirley Jowell and Peter F. Patel-Schneider. It was subsequently modified by Jennifer Ballentine and Thomas Dean, Bernhard Nebel, Daniel Pagenstecher, Kurt Steinkraus, Toby Walsh and Carles Sierra. The current version has been prepared by Marc Pujol-Gonzalez, Francisco Cruz-Mencia, and Diego Calvanese.

# References

Abelson, H.; Sussman, G. J.; and Sussman, J. 1985. *Structure and Interpretation of Computer Programs*. Cambridge, Massachusetts: MIT Press.

Baumgartner, R.; Gottlob, G.; and Flesca, S. 2001. Visual information extraction with Lixto. In *Proceedings of the 27th International Conference on Very Large Databases*, 119–128. Rome, Italy: Morgan Kaufmann.

Brachman, R. J., and Schmolze, J. G. 1985. An overview of the KL-ONE knowledge representation system. *Cognitive Science* 9(2):171–216.

Gottlob, G.; Leone, N.; and Scarcello, F. 2002. Hypertree decompositions and tractable queries. *Journal of Computer and System Sciences* 64(3):579–627.

Gottlob, G. 1992. Complexity results for nonmonotonic logics. *Journal of Logic and Computation* 2(3):397–425.

Levesque, H. J. 1984a. Foundations of a functional approach to knowledge representation. *Artificial Intelligence* 23(2):155–212.

Levesque, H. J. 1984b. A logic of implicit and explicit belief. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, 198–202. Austin, Texas: American Association for Artificial Intelligence.

Nebel, B. 2000. On the compilability and expressive power of propositional planning formalisms. *Journal of Artificial Intelligence Research* 12:271–315.

1. [↑](#footnote-ref-1)
2. This is how your footnotes should appear. [↑](#footnote-ref-2)
3. Note the line separating these footnotes from the text. [↑](#footnote-ref-3)