Chapter 1 : Prolog and Natural-Language Analysis : Download Free Book

ming language Prolog by way of example programs that apply it to the problem of natural-language analysis and processing. This volume began as the notes for a tutorial taught by one of the authors.

Clauses with bodies are called rules. An example of a rule is: If we add that rule and ask what things are animals? For this reason, a comparatively small set of library predicates suffices for many Prolog programs. These predicates are not given a relational meaning and are only useful for the side-effects they exhibit on the system. Logically, the Prolog engine tries to find a resolution refutation of the negated query. The resolution method used by Prolog is called SLD resolution. If the negated query can be refuted, it follows that the query, with the appropriate variable bindings in place, is a logical consequence of the program. In that case, all generated variable bindings are reported to the user, and the query is said to have succeeded. In that case, the system creates a choice-point, unifies the goal with the clause head of the first alternative, and continues with the goals of that first alternative. If any goal fails in the course of executing the program, all variable bindings that were made since the most recent choice-point was created are undone, and execution continues with the next alternative of that choice-point. This execution strategy is called chronological backtracking. This results in the following query being evaluated as true: Yes This is obtained as follows: Initially, the only matching clause-head for the query sibling sally, erica is the first one, so proving the query is equivalent to proving the body of that clause with the appropriate variable bindings in place, i. The next goal to be proved is the leftmost one of this conjunction, i. Two clause heads match this goal. Again, this can be proved by the corresponding fact. Since all goals could be proved, the query succeeds. Since the query contained no variables, no bindings are reported to the user. A query with variables, like: Notice that with the code as stated above, the query? One would insert additional goals to describe the relevant restrictions, if desired. Prolog attempts to prove illegal X. If a proof for that goal can be found, the original goal i. If no proof can be found, the original goal succeeds. This kind of negation is sound if its argument is "ground" i. Soundness is lost if the argument contains variables and the proof procedure is complete. In particular, the query? Programming in Prolog[edit] In Prolog, loading code is referred to as consulting. Prolog can be used interactively by entering queries at the Prolog prompt? If there is no solution, Prolog writes no. If a solution exists then it is printed. If there are multiple solutions to the query, then these can be requested by entering a semi-colon ;. There are guidelines on good programming practice to improve code efficiency, readability and maintainability. An example of a query: As an example, an optimizing compiler with three optimization passes could be implemented as a relation between an initial program and its optimized form: Quicksort[edit] The quicksort sorting algorithm, relating a list to its sorted version: Design patterns[edit] A design pattern is a general reusable solution to a commonly occurring problem in software design. In Prolog, design patterns go under various names: Higher-order logic and Higher-order programming A higher-order predicate is a predicate that takes one or more other predicates as arguments. This can be used for list comprehension. For example, perfect numbers equal the sum of their proper divisors: This can be used to enumerate perfect numbers, and also to check whether a number is perfect. As another example, the predicate maplist applies a predicate P to all corresponding positions in a pair of lists: Modules[edit] For programming in the large , Prolog provides a module system. The module system is standardised by ISO. Most notably, the rewriting equips the predicate with two additional arguments, which can be used to implicitly thread state around,[clarification needed] analogous to monads in other languages. DCGs are often used to write parsers or list generators, as they also provide a convenient interface to difference lists. Meta-interpreters and reflection[edit] Prolog is a homoiconic language and provides many facilities for reflection. Its implicit execution strategy makes it possible to write a concise meta-circular evaluator also called meta-interpreter for pure Prolog code: Since Prolog programs are themselves sequences of Prolog terms: For example, Sterling and Shapiro present a meta-interpreter that performs reasoning with uncertainty, reproduced here with slight modifications: This interpreter uses a table of built-in Prolog predicates of the form [29]: Given those, it can be called as solve Goal, Certainty to execute Goal and obtain a measure of certainty about the result. Turing completeness[edit]

Pure Prolog is based on a subset of first-order predicate logic , Horn clauses , which is Turing-complete. Turing completeness of Prolog can be shown by using it to simulate a Turing machine: A simple example Turing machine is specified by the facts: This machine performs incrementation by one of a number in unary encoding: It loops over any number of "1" cells and appends an additional "1" at the end. Example query and result:

DOWNLOAD PDF PROLOG AND NATURAL-LANGUAGE ANALYSIS

Chapter 2 : CiteSeerX â€" Citation Query Prolog and natural-language analysis: Into the third decade

This digital edition of Prolog and Natural-Language Analysis is distributed at no charge for noncommercial use by Microtome Publishing. Contents.

Although the class of standard CCGs is known to be polynomially parsable, use of variables suggests more complexity fo Although the class of standard CCGs is known to be polynomially parsable, use of variables suggests more complexity for processing GTRCs. First, we show that an experimental parser runs polynomially in practice on a realistic fragment of Japanese by eliminating spurious ambiguity and excluding genuine ambiguities. This paper presents a robust parsing approach which is designed to address the issue of syntactic errors in text. The approach is based on the concept of an error grammar which is a grammar of ungrammatical sentences. An error grammar is derived from a conventional grammar on the basis of An error grammar is derived from a conventional grammar on the basis of an analysis of a corpus of observed ill-formed sentences. A robust parsing algorithm is presented which is applied after a conventional bottom-up parsing algorithm has failed. This algorithm combines a rule from the error grammar with rules from the normal grammar to arrive at a parse for an ungrammatical sentence. This algorithm is applied to 50 test sentences, with encouraging results. Show Context Citation Context Add a word For each rule in the grammar excluding rules expanding pre-terminal symbols , an error rul Memo functions and memoization are well known concepts in AI programming. They have been around since the sixties and are often used as examples in introductory programming texts. However, the automation of memoization as a practical software engineering tool for AI systems has never received However, the automation of memoization as a practical software engineering tool for AI systems has never received a detailed treatment. This paper describes how automatic memoization can be made viable on a large scale. It points out advantages and uses of automatic memoization not previously described, describes the components of an automatic memoization facility, enumerates potential memoization failures, and presents a publicly available memoization package CLAMP for the Lisp programming language. Experience in applying these techniques in the development of a large planning system are briefly discussed. AI programming, AI software engineering. This work was supported in part by the Advanced Research Projects Agency This paper provides a set of criteria and guidelines for grammar design, which are illustrated by an implemented grammar fragment for German including semantic composition rules. The connection between strings, syntactic categories, and semantic representations is realized as a mathematical relati The connection between strings, syntactic categories, and semantic representations is realized as a mathematical relation. This is a basis for the reversibility of grammars, which is essential for the efficient use of grammatical ressources in Machine Translation systems and for the verification of grammars in general. Relational approaches nowadays are called constraint-based approaches, where the various modules of a grammar are seen as sets of constraints which contribute to the intensional description of the set of legal string-syntax-semantics triples. Obviously, grammar design relies heavily on the linguistic theory and the programming language which is used for the implementation. The implementation formalism is the CUF-language, which is an extension of Prolog by feature terms, types, and a sophisticated evaluation strategy.

Chapter 3 : Prolog and Natural-Language Analysis - Digital Edition

Prolog and Natural Language Analysis provides a concise and practical introduction to logic programming and the logic-programming language Prolog both as vehicles for understanding elementary computational linguistics and as tools for implementing the basic components of natural-language-processing systems.

Chapter 4 : CiteSeerX â€" Citation Query Prolog and Natural-Language Analysis, CSLI Lecture Notes, Nr 1

Free Prolog ebook "Prolog and Natural-Language Analysis" by Fernando C. N. Pereira and Stuart M. Shieber in pdf

format.. Description. This book provides, in one volume, one of the best introductions to prolog programming and one of the best introductions to natural language processing.

Chapter 5 : Microtome Publishing

His research on Prolog and natural-language processing underlies much recent work in logic grammars. Stuart Shieber is a researcher at the Center for the Study of Language and Information and a computer scientist at SRI International's Artificial Intelligence Center.

Chapter 6 : Prolog and Natural-Language Analysis - Download link

"Prolog and Natural Language Analysis" provides a concise and practical introduction to logic programming and the logic-programming language Prolog both as vehicles for understanding elementary computational linguistics and as tools for implementing the basic components of natural-language-processing systems.

Chapter 7 : Prolog - Wikipedia

Prolog and Natural-Language Analysis - Digital Edition Fernando C. N. Pereira and Stuart M. Shieber This work is covered by copyright, which restricts the uses that you may make of the work.