Project proposals

May 27, 2017

1 AI course - Project proposals

1.1 Homework 1

Considering the DPLL implementation developed in class, extend it with pure literal detection and propagation. Perform experimental analysis with random 3-CNFs confirming/disconfirming that pure literal propagation is effective.

1.2 Homework 2

Considering the DPLL implementation developed in class, extend it with the "Maximum Occurrences in clauses of Minimum Size" (MOMS) heuristic. The heuristic works by preferring literals occurring often in short clauses. Ties are broken randomly. Perform experimental analysis with random 3-CNFs confirming/disconfirming that MOMS is more effective than random variable selection.

1.3 Homework 3

Use DPLL to reason about logical implication of non-CNF formulas. Formulas are written as lists, with strings 'AND', 'OR' and 'NOT' standing for symbols $\wedge$, $\vee$ and $\neg$, respectively. Integers are used to encode propositions, e.g., 1 stands for p_1, 2 for p_2 and so on. For instance the formula $(p_1 \wedge p_2) \vee \neg p_3$ is represented as

```
In [2]: formula = ['OR', ['AND', 1, 2], ['NOT', 3]]
        print(formula)

['OR', ['AND', 1, 2], ['NOT', 3]]
```

The goal is to check if a given formula φ is logical consequence of a theory Γ, i.e., a list of formulas. This is true only if $\Gamma \wedge \neg\varphi$ is unsatisfiable.

1.4 Homework 4

Use DPLL to reason about the equivalence of non-CNF formulas. Formulas are represented as above. Given two formulas φ_1 and φ_2, the goal is to check whether $\varphi_1 \oplus \varphi_2$ is satisfiable or not, where $\oplus$ stands for the "exclusive or" operator.

1.5 Homework 5

Pick some common verbs and complete the following tasks:

Write a program to find those verbs in the Prepositional Phrase Attachment Corpus nltk.corpus.ppattach. Find any cases where the same verb exhibits two different attachments, but where the first noun, or second noun, or preposition, stay unchanged.

Devise CFG grammar productions to cover some of these cases.

1.6 Homework 6

Write a program to compare the efficiency of a top-down chart parser compared with a recursive descent parser. Use the same grammar and input sentences for both. Compare their performance using the time module.

1.7 Homework 7

Download some electronic books from Project Gutenberg. Write a program to scan these texts for any extremely long sentences. What is the longest sentence you can find? What syntactic construction(s) are responsible for such long sentences?

1.8 Homework 8

Process each tree of the Treebank corpus sample nltk.corpus.treebank and extract the productions with the help of Tree.productions(). Discard the productions that occur only once. Productions with the same left hand side, and similar right hand sides can be collapsed, resulting in an equivalent but more compact set of rules. Write code to output a compact grammar.

```
In [ ]:
```