Benjamin Fenster
CS-251 (Wu)
19 December 2005

Notes – Extra Topics

I. Functional Languages
  a. Motivations
    i. Should be following the mathematical way of doing functions
    ii. In Prolog, have many *dependent* predicates
    iii. In functional languages, evaluate all independently.
    iv. Should not be influenced by the sequence of code (though in practice it is)
    v. Mathematical functions do not produce side effects (again, not true in practice)
  b. Definition
    i. In imperative languages, one has strong restrictions on what can be returned and cannot be returned from functions.
    ii. There are only two types of data: atom, list
      1. Atoms: All symbols and numeric constants, just like Prolog
      2. Lists
        a. Separated by delimiting their elements with parentheses
        b. (A (A B) D (E (F G)))
      3. The empty list is both an atom and a list
    iii. Must Have:
      1. Starting functions (primitive / built-in functions)
      2. A set of functional forms to build more advanced functions
        a. Either takes functions as parameters, yields a function as a result, or both
        b. Function composition (f ° g) is a common functional form
      3. Function application operator (either an interpreter or compiler)
    iv. Everything in LISP (atoms and lists) are called S-expressions (λ)
  c. LISP
    i. Primitives
      1. QUOTE returns its parameters as-is
        a. (QUOTE (A B C)) ⇓ (A B C)
        b. ‘A is equivalent to (QUOTE A)
      2. CAR is similar to head in prolog
        a. (CAR ‘((A B) C D)) ⇓ (A B)
        b. The quote keeps it from trying to evaluate ((A B) C D) as a function and instead just treats it as-is
      3. CDR Is like tail. (CDR ‘(A)) ⇓ ()
      4. CONS is for list construction
        a. (CONS ‘A ‘(C B)) ⇓ (A C B)
        b. (CONS ‘(C D) ‘(A B)) ⇓ (C D A B)
      5. EQ
        a. Takes two parameters
        b. Returns t if they’re equal
        c. Returns nil if they’re not equal
      6. ATOM: Returns t iff its argument is an atom
      7. NULL: Returns t iff its argument is the empty list
      8. EVAL: Is the application operation (as noted)
      9. These are available in all LISP systems
    ii. Lambda Expressions
      1. A lambda expression: func_name (LAMBDA (a b … n) map_function)
      2. lambda (x) x * x * x (2) ⇓ (8)
      3. Note that the declaration discusses the input (x) but says nothing about the output. The output is just whatever result the function gives
  d. History
    i. LISP
      1. Designed by John McCarthy at MIT (1958- 1959)

            2. A member of the National Academy of Sciences, won the Turing award, supervised only 22 graduate students (including Masters students)
            3. Very serious; very strict
        ii. Scheme: Small, static-scoped LISP descendent
        iii. Common LISP: An amalgam of various dialects of LISP
    e. EMACS LISP
        i. Always running in emacs
        ii. Type (+ 47 38)
        iii. Put the cursor after the closing parenthesis.
        iv. Type C-x C-e
        v. The result goes in the message mini-buffer
    f. LISP vs. Prolog
        i. MIT and Stafford use LISP
        ii. Duke and Europe use Prolog
        iii. This isn't really for any academic reasons; just historical
        iv. Prolog is more user friendly and easier to write small scaled application programs
        v. LISP provides good facilities to design heuristic inference engines
        vi. "Probably 85 of the 100 best-known programs in AI would be in LISP" – Charniak and McDermott, 1985

II. Natural Language Processing
    a. Modern view says speech is a form of action (as opposed to logic, which says there are only true/false statements)
    b. Goals
        i. Inform
        ii. Query
        iii. Command
        iv. Promise
        v. Acknowledge
    c. Stages
        i. Intention (speaker S wants to inform hearer H that P)
        ii. Generation (S selects words W to express P)
        iii. Synthesis (S utters W)
        iv. Perception (H perceives W')
        v. Analysis (H infers meanings $P_1, \ldots, P_n$)
        vi. Disambiguation (H infers intended meaning P_i)
        vii. Incorporation (H incorporates P into the knowledge base)
    d. Problems
        i. Insincerity (S doesn't believe P)
        ii. Speech wreck ignition failure (Note: This probably means something, but nobody happened to offer any explanation)
        iii. Ambiguous utterance
        iv. Differing understandings of the current situation
            1.
    e. Grammar
        i. Taking the pre-1958 view
        ii. Grammar is a set of rewrite rules (S → NP VP) for sentence S (is Very Phrase followed by Noun Phrase). See [CMSI-164] for some details.
        iii. Language is a set of strings of terminal symbols (Article → a | an | the)
        iv. Types: Regular, Context-Free, Context-Sensitive. See [CS-243] for details.
        v. Most natural languages are context free and parsable in real time.
    f. Grammaticality Judgments
        i. Formal language L_1 may differ from natural language L_2
        ii. False positives: Sentences that exist in L_1 even though they're not really allowed in the natural language
        iii. False negatives: Sentences that don't exist in L_1 even though they are allowed

        iv. It becomes a learning problem to get them to agree and ambiguity may demand leaving deliberate differences
        v. Real grammars range from 10 to 500 pages and still aren't sufficient for English.
        vi. Efficient algorithms $O(n^3)$ run at several thousand words per second for real grammars
    g. Applying Prolog to Natural Language Processing
        i. Syntactic vs. Semantic Approaches
            1. Syntactic: Based solely on rules. Given rules, understand the sentence.
            2. Semantic: Given some background understanding at the concept level
        ii. Stages
            1. Parsing: Analyze the syntactic structure of the sentence
            2. Semantic Interpretation
            3. Contextual/Real-World interpretation

III. Semantic Web
    a. World Wide Web today relies on textual searches
    b. Websites don't yet support any kind of *semantic* search (by meaning)
    c. Information on the web is designed for human consumption
    d. The semantic web approach develops languages that can be processed by machine
    e. Generations of the World Wide Web
        i. First: Static, hand-written pages
        ii. Second: Can generate pages from user interaction (where we are now)
        iii. Third: Semantic
    f. This is an initiative of the W3C
    g. Example
        i. Want the cheapest copy (including shipping) of a particular book that can be obtained within one week
        ii. One-World Mediation: All book sites are in the same world
    h. Example
        i. A home buyer wants a house for under $price in a neighborhood with a school in the top third…
        ii. Multi-world mediation
        iii. Requires expert knowledge in multiple domains
        iv. Complex Multiple Worlds
    i. RDF
        i. Add annotations for websites
        ii. Agent-Oriented Languages
            1. Like object oriented programming in that it's a new design system
            2. Give the agent beliefs, ideas, principles
            3. Based on those beliefs it will make a decision.
            4. Agent-O is one example
        iii. Have a RDF repository on top of the ordinary HTML/GIF/JPEG content.
        iv. Agents communicate only with the repository
        v. The first semantic website: owl.mindswap.org