Logic Programming using LISP

Kunal Banerjee

(Teaching Assistant)

Discrete Structures (CS21001)

Computer Science & Engineering Department

Indian Institute of Technology, Kharagpur

November 12, 2013

Outline

LISP History

- LISP = LISt Processing
- Invented by *John McCarthy* in 1958
- Implemented Church's lambda-calculus (recursive function theory)
- Widely used for applications in AI ever since
- Many versions: Franz, Mac, Inter, Common

Evaluating Simple Expressions

```
> -3
-3
> “Hello, World!”
“Hello, World!”
> t      ;; LISP is case-insensitive
T
> nil    ;; t represents “true”, and nil represents “false”
NIL
```

Evaluating Lists as Functions

Lisp program code takes the form of lists.
A *list* begins with a parenthesis, then immediately contains a symbol, then zero or more expressions separated with white space, then a closing parenthesis.

> (+ 3 2 7 9)

21

> (* 4 2.3) ;; not typed

9.2

> (length “Four score and seven years ago”)

30

> (< 3 9) ;; Is 3 < 9 ?

T

> (numberp “hello”) ;; Is “hello” a number ?

NIL

Control Structures and Variables

```
> (if (<= 3 2) (* 3 9) (+ 4 2 3))    ;; if 3 <= 2 then return 3*9 else return 4+2+3
  9
> (setf x (* 3 2))
  6
> x
  6
> (setf y (+ x 3))
  9
> (dotimes (z 3 "world") (print "hello"))
  hello
  hello
  hello
  world
```

Writing Functions

```
> (defun add-four (x) (+ x 4))
ADD-FOUR
> (add-four 7)
11
> (defun factorial (n)
    (if (<= n 0)
      1
      (* n (factorial (- n 1)))))
FACTORIAL
> (factorial 4)
24
```

Problems in (Propositional) Logic

All examples are taken from the tutorials presented in the course *Logics for Computer Science (CS60031)*.

- Murder Mystery
- Deciding Curriculum
- A Game of Chess
- Prices of Commodity

Murder Mystery : Problem

There are three suspects for a murder: Adams, Brown and Clark.
Adams says “I didn’t do it. The victim was an old acquaintance of Brown’s, but Clark hated him.”
Brown says “I didn’t do it. I didn’t even know the guy. Besides, I was out of town all that week.”
Clark says “I didn’t do it. I saw both Adams and Brown downtown with the victim that day; one of them must have done it.”
Assume that the two innocent men are telling the truth, but the guilty man might not be.

Who did it?

Murder Mystery : Solution

Let us consider the following propositional constants:
A: Adams tells the truth, B: Brown tells the truth, C: Clark tells the truth.
Since two innocent men are telling the truth, of the 8 interpretations, the set of possible ones are $\{\langle A, B, C\rangle = \langle F, T, T\rangle, \langle T, F, T\rangle, \langle T, T, F\rangle, \langle T, T, T\rangle\}$.
Adams' contention is in conflict with Brown's because Adams said, "... The victim was an old acquaintance of Brown's ...", whereas Brown said, "... I didn't even know the guy...". So, the interpretations $\langle T, T, F\rangle$ and $\langle T, T, T\rangle$ are ruled out.
Similarly, Brown's contention "... I didn't even know the guy..." is in conflict with Clark's "I saw both Adams and Brown downtown with the victim that day". So, the interpretation $\langle F, T, T\rangle$ is also ruled out.
Thus, we are left with only one interpretation $\langle T, F, T\rangle$. Under this interpretation, all the statements made by Adams and those made by Clark are true and that creates no conflict with each other.
Thus, it's **Brown who is lying and hence is the murderer**.

Murder Mystery : Code

```
> (defun sentence (A B C) (and (or (and A (not B)) (and (not A) B))
      (or (and B (not C)) (and (not B) C)) ))
  SENTENCE
> (if (sentence NIL T T) (print "A"))
  NIL
> (if (sentence T NIL T) (print "B"))
  "B"
  "B"    ;; Why is the second "B" printed ?
> (if (sentence T T NIL) (print "C"))
  NIL
> (if (sentence T T T) (print "Dont Know!"))
  NIL
```

Murder Mystery : Code 2 – Making the LISP Compiler work a little more

```
> (defun sentence (A B C) (and (or (and A (not B)) (and (not A) B))
    (or (and B (not C)) (and (not B) C)) ))
SENTENCE
> (dolist (A '(t nil))
   (dolist (B '(t nil))
    (dolist (C '(t nil))
     (if (or (and A B) (and B C) (and C A))
      (if (sentence A B C)
       (if (not A) (print "A")
        (if (not B) (print "B")
         (if (not C) (print "C")))))))))
"B"
NIL
```

Deciding Curriculum : Problem

Encode the following arguments and show whether they are valid or not. If not valid, give counter-models, i.e., truth assignments to the propositions which make them false.

If algebra is required or geometry is required, then all students will study mathematics.
Algebra is required and trigonometry is required.
Therefore, all students will study mathematics.

Deciding Curriculum : Encoding

Propositions

Algebra:	Algebra is required
Geometry:	Geometry is required
Trigonometry:	Trigonometry is required
StudyMaths:	All students will study mathematics

Given constraints and goal
C1: Algebra $\vee$ Geometry $\Rightarrow$ StudyMaths
C2: Algebra $\wedge$ Triginometry
Goal: StudyMaths

We need to show either

$$(C1 \wedge C2) \Rightarrow G \quad \text{is valid, or}$$
$$(C1 \wedge C2 \wedge \neg G) \quad \text{is unsatisfiable}$$

Deciding Curriculum : Code

```
> (defun imply (1st 2nd) (or 2nd (not 1st)))
IMPLY
> (setf Algebra T)
T
> (setf Trigonometry T)
T
> (dolist (Geometry '(t nil))
    (dolist (StudyMaths '(t nil))
      (if (and (imply (or Algebra Geometry) StudyMaths) Algebra Trigonometry (not
StudyMaths))
          (progn
          (print "Algebra") (print Algebra)
          (print "Geometry") (print Geometry)
          (print "Trigonometry") (print Trigonometry)
          (print "Study Maths") (print StudyMaths)))))
NIL ;; Therefore, all students will indeed study mathematics
```

A Game of Chess : Problem

Encode the following arguments and show whether they are valid or not. If not valid, give counter-models, i.e., truth assignments to the propositions which make them false.

(a) *If the king does not castle and the pawn advances, then the bishop is blocked or the rook is pinned.*

(b) *If the king does not castle, then if the bishop is blocked, then the game is a draw.*

(c) *The king castles or if the rook is pinned, then the exchange is lost.*

(d) *The king does not castle and the pawn advances.*

(e) *Therefore, the game is a draw or the exchange is lost.*

A Game of Chess : Encoding

Propositions

K:	King castles	P:	Pawn advances
B:	Bishop is blocked	R:	Rook is pinned
G:	Game is a draw	E:	Exchange is lost

Given constraints and goal

C1: $(\neg K \wedge P) \Rightarrow (B \vee R)$

C2: $\neg K \Rightarrow (B \Rightarrow G)$

C3: $K \vee (R \Rightarrow E)$

C4: $\neg K \wedge P$

Goal: $G \vee E$

A Game of Chess : Code

```
> (dolist (K '(t nil))
    (dolist (P '(t nil))
      (dolist (B '(t nil))
        (dolist (R '(t nil))
          (dolist (G '(t nil))
            (dolist (E '(t nil))
              (if (and
                (imply (and (not K) P) (or B R))    ;; Constraint 1
                (imply (not K) (imply B G))    ;; Constraint 2
                (or K (imply R E))    ;; Constraint 3
                (and (not K) P)    ;; Constraint 4
                (not (or G E)))    ;; Negated Goal
                  (progn
                  (print "King castles") (print K)
                  (print "Pawn advances") (print P)
                  (print "Bishop is blocked") (print B)
                  (print "Rook is pinned") (print R)
                  (print "Game is a draw") (print G)
                  (print "Exchange is lost") (print E) ))))))))
NIL
```

Prices of Commodity : Problem

Encode the following arguments and show whether they are valid or not. If not valid, give counter-models, i.e., truth assignments to the propositions which make them false.

(a) *If a scarcity of commodities develops, then the prices rise.*

(b) *If there is a change of government, then fiscal controls will not be continued.*

(c) *If the threat of inflation persists, then fiscal controls will be continued.*

(d) *If there is overproduction, then prices do not rise.*

(e) *Either there is overproduction or there is a change of government.*

(f) *Therefore, either a scarcity of commodities does not develop or there is a change of government.*

Prices of Commodity : Encoding

Propositions

S:	Scarcity develops	P:	Prices rise
G:	Government changes	F:	Fiscal controls continued
I:	Inflation persists	O:	Overproduction occurs

Given constraints and goal

C1: $S \Rightarrow P$

C2: $G \Rightarrow \neg F$

C3: $I \Rightarrow F$

C4: $O \Rightarrow \neg P$

C5: $(O \wedge \neg G) \vee (\neg O \wedge G)$

Goal: $(\neg S \wedge \neg G) \vee (S \wedge G)$

Prices of Commodity : Code

```
> (dolist (S '(t nil))
    (dolist (P '(t nil))
      (dolist (G '(t nil))
        (dolist (F '(t nil))
          (dolist (I '(t nil))
            (dolist (O '(t nil))
              (if (and
                (imply S P)    ;; Constraint 1
                (imply G (not F))    ;; Constraint 2
                (imply I F)    ;; Constraint 3
                (imply O (not P))    ;; Constraint 4
                (or (and O (not G)) (and (not O) G))    ;; Constraint 5
                (not (or (and (not S) (not G)) (and S G))))    ;; Negated Goal
                (progn
                (print "Scarcity develops") (print S)
                (print "Prices rise") (print P)
                (print "Change of government") (print G)
                (print "Fiscal controls continued") (print F)
                (print "Threat of inflation persists") (print I)
                (print "Overproduction occurs") (print O) ))))))))
```

Prices of Commodity : Code (contd.)

"Scarcity develops"

NIL

"Prices rise"

T ;; Also for "Prices rise" = NIL

"Change of government"

T

"Fiscal controls continued"

NIL

"Threat of inflation persists"

NIL

"Overproduction occurs"

NIL

Applications of LISP

- Mirai – A 3D graphics suite
- Prototype Verification System (PVS) – A mechanized environment for formal specification and verification
- ACL2 – An automated theorem prover
- Axiom, Maxima – Computer algebra systems
- and many more . . .

Resources for LISP

Download from `http://www.clisp.org/`

Tutorials available at

`http://cs.gmu.edu/~sean/lisp/LispTutorial.html`
`http://en.wikibooks.org/wiki/Common_Lisp/First_steps/Beginner_tutorial`
`http://www.cs.sfu.ca/CourseCentral/310/pwfong/Lisp/1/tutorial1.html`

Thank You !!!

kunalb@cse.iitkgp.ernet.in