**COMP 3710 Applied Artificial Intelligence**

**Seminar/Lab 6.**

**Backward Chaining for ZooKeeper**

1. **Objectives**

* Use of backward chaining
* How to convert rules and facts to Horn forms
* How to use DNF in the implementation of backward chaining

1. **Exercise**

rules['Mammal'] = [['Hair'], ['Milk']];

rules['Ungulate'] = [['Mammal', 'Hoof'], ['Mammal', 'ChewCud']];

rules['Giraffe'] = [['Ungulate', 'LongLeg', 'LongNeck', 'TawnyColor', 'DarkSpot']];

rules['Zebra'] = [['Ungulate', 'WhiteColor', 'BlackStrip']];

facts = ['ChewCud', 'DarkSpot', 'Hair', 'LongLeg', LongNeck', 'TawnyColor'];

Let’s find if that animal is giraffe, using backward chaining.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| BC(…) | Fact? | No rule? | Rule0 | Rule1 | Return |
| BC(‘Giraffe’) | No | No | [‘Ungulate’, ‘LongLeg’, ‘LongNeck’, ‘TawnyColor’, ‘DarkSpot’] |  | BC(‘Ungulate’) && BC(‘LongLeg’) && BC(‘LongNeck’) && BC(‘TawnyColor’) && BC(‘DarkSpot’) |
| BC(‘Ungulate’) | No | No | [‘Mammal’, ‘Hoof’] | [‘Mammal’, ‘ChewCud’] | BC(‘Mammal’) && BC(‘Hoof’) **||** BC(‘Mammal’) && BC(‘ChewCud’) |
| BC(‘Mammal’) | No | No | [‘Hair’] | [‘Milk’] | BC(‘Hair’) || BC(‘Milk’) |
| BC(‘Hair’) | Yes |  |  |  | TRUE |
| BC(‘Milk’) | No |  |  |  | FALSE |
| BC(‘Mammal’) |  |  |  |  | TRUE || FALSE => **TRUE** |
| BC(‘Hoof’) | No |  |  |  | FALSE |
| BC(‘Mammal’) | **Yes** |  |  |  | TRUE |
| BC(‘ChewCud’) | Yes |  |  |  | TRUE |
| BC(‘Ungulate’) |  |  |  |  | TRUE && FALSE || TRUE && TRUE => **TRUE** |
| BC(‘LongLeg’) | Yes |  |  |  | TRUE |
| BC(‘LongNeck’) | Yes |  |  |  | TRUE |
| BC(‘TawnyColor) | Yes |  |  |  | TRUE |
| BC(‘DarkSpot’) | Yes |  |  |  | TRUE |
| BC(‘Giraffe’) |  |  |  |  | TRUE && TRUE && TRUE && TRUE && TRUE => **TRUE** |

Your work:

Try the above simulation with ‘Zebra’.

1. **Algorithm**

// rules['Bird'] = [['Feather'], ['Fly', 'LayEgg']];

// facts = ['ChewCud', 'DarkSpot', 'Hair', 'LongLeg', LongNeck', 'TawnyColor'];

*ps* = ‘Bird’; // Example

Algorithm of *backwardChining*(*ps*):

If ??? is a fact,

Return ture;

If there is no more rule to use for ???,

Return flase;

// Now you need to use backward chaining again for each rule.

For the rules used for *ps*, *valid* = *backwardChaining*(rules[ps][0][0]) ||

(*backwardChaining*(rules[ps][1][0]) && *backwardChaining*(rules[ps][1][1]));

// You need to generalize this step.

// array\_variable.length can be used to check the number of elements in an array.

If (*valid*)

Make *ps* a fact;

Return ???;

1. **How to implement backward chaining for ZooKeeper**

The data structures used in BC:

var rules = {};

rules['Mammal'] = [['Hair'], ['Milk']]; // rules for Mammal

...

var facts = [];

facts[0] = 'Hair';

...

var animals = ['Albatross', 'Penguin', 'Ostrich', 'Zebra', 'Giraffe', 'Tiger',

'Cheetah'];

The functions that you can use:

function isFact(ps) // ps: propositional symbol

function pushIntoFacts(ps)

function thereIsNoRuleToUse(ps)

function print\_msg(msg), println\_msg(msg), clear\_msg() // utility functions

// to display messages

The functions that you need to implement:

* + function **find**()

When the user clicks the button, this function will be invoked.

This function should print a proper message for each animal.

* + function **backwardChaining**(ps) // ps: propositional symbol

You can start with

* w6\_backward\_chaining\_zookeeper\_student.html

1. **Something to think**

* What if there are more facts?

1. **Assignment**
   * The title of the mail should include your name, id, and COMP 3710.

* You need to submit a document file for 2), not image of hand-written document.
  + Total marks: 5
  + Due: 6:00 PM, October 20, 2017
* You need to submit the application in 4) by email, with the entire screen shot, not just window content, that shows how your application works. (Any submission with a window content shot will not be accepted.)
  + **Please do not share the code.** Any violation could cause a very serious problem.
  + Total marks: 10
    - * Any syntax error: 0 mark
      * No completion: 0 mark
        + Generally no partial marks for any code that cannot find a solution. You really need to complete this programming.
  + Due: 6:00 PM, October 25, 2017
* Any late submission will NOT be accepted.