**COMP 3710 Applied Artificial Intelligence**

**Seminar/Lab 7.**

**Backward Chaining for *ZooKeeper***

1. **Objectives**
* Use of backward chaining for expert system
* How to convert rules and facts to Horn forms
* How to use DNF in the implementation of backward chaining
1. **Exercise 1**

// DNF for rules

// rules: associative array in which property values are 2D arrays.

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

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

 // Two rules can be used for Ungulate.

 // if Mammal **and** Hoof, then Ungulate, **or**

 // if Mammal **and** ChewCud, then Ungulate

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

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

// Linear array

// This array starts with the facts obtained by observing an animal.

facts = ['ChewCud', 'BlackStrip', 'Hair', 'WhiteColor'];

Let’s find if that animal is zebra, using backward chaining – **recursion**.

|  |  |  |
| --- | --- | --- |
| Facts | The goals to check | Matching rules |
| ChueCud, BlackStrip, Hair, WhiteColor | Zebra(?) | Zebra - [Ungulate, WhiteColor, BlackStrip] |
| ChueCud, BlackStrip, Hair, WhiteColor | Ungulate(?) && WhiteColor(?) && BlackStrip(?) | Ungulate - [Mammal, Hoof], [Mammal, ChewCud] |
| ChueCud, BlackStrip, Hair, WhiteColor | (Mammal(?) && Hoof(?) || Mammal(?) && ChewCud(?)) && WhiteColor(?) && BlackStrip(?) | Mammal - [Hair], [Milk] |
| ChueCud, BlackStrip, Hair, WhiteColor | ((Hair(?) || Milk(?)) && Hoof(?) || Mammal(?) && ChewCud(?)) && WhiteColor(?) && BlackStrip(?) | Hair - TRUE |
| ChueCud, BlackStrip, Hair, WhiteColor,**Mammal** | (TRUE && Hoof(?) || Mammal(?) && ChewCud(?)) && WhiteColor(?) && BlackStrip(?) | Hoof - FALSE |
| ChueCud, BlackStrip, Hair, WhiteColor,Mammal | (FALSE || Mammal(?) && ChewCud(?)) && WhiteColor(?) && BlackStrip(?) | Mammal - TRUE |
| ChueCud, BlackStrip, Hair, WhiteColor,Mammal | (FALSE || TRUE && ChewCud(?)) && WhiteColor(?) && BlackStrip(?) | ChewCud – TRUE |
| ChueCud, BlackStrip, Hair, WhiteColor,Mammal | TRUE && WhiteColor(?) && BlackStrip(?) | WhiteColor – TRUE  |
| ChueCud, BlackStrip, Hair, WhiteColor,Mammal | TRUE && TRUE && BlackStrip(?) | BlackStrip – TRUE  |
| ChueCud, BlackStrip, Hair, WhiteColor,Mammal | TRUE && TRUE && TRUE | BlackStrip – TRUE  |
| ChueCud, BlackStrip, Hair, WhiteColor,Mammal | TRUE |  |

Your work: Try the above simulation for ‘Giraffe’ with facts = ['ChewCud', 'DarkSpot', 'Hair', 'LongLeg', LongNeck', 'TawnyColor'].

1. **Algorithm of backward chaining**

// E.g.,

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

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

// rules['Bird'][0][0] is 'Feather', and

// rules['Bird'][1][0] is 'Fly' and rules['Bird'][1][1] is 'LayEgg'.

// if (['Bird'][0][0] in facts **||** (['Bird'][1][0] in facts **&&** ['Bird'][1][1] in facts)),

// then 'Bird' is a fact.

*ps* = 'Bird'; // Example

Algorithm of *BC*(*rules*, *facts*, *ps*): // Check if *ps* is a fact

If ??? is a fact,

 Return ???;

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

 Return ???;

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

 ???

For each rule used for *ps*,

 *valid* = *valid* || *BC*\_*anded*(*rules*, *facts*, **???**);

 // The backward chaining results for multiple rules should be ORed.

 // The backward chaining results for multiple propositional symbols in a rule

 // should be ANDed.

 If (*valid*)

 Make *ps* a fact;

 Return ???;

1. **Exercise 2**

Complete the posted exercise program.

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';

...

// animal candidates

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

The functions that you can use:

function printMessage(msg), printlnMessage(msg), clearMessage() // utility functions to display messages

The functions that you need to complete:

* + function **find**()

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

This function should print a proper message for each animal.

* + function **BC**(rules, facts, ps) // ps: propositional symbol

Check if ps is valid, i.e., a fact

* + function **BC\_anded**(rules, facts, rule)

Check if a rule is valid

* function is\_fact(facts, ps) // ps: propositional symbol

Return true or false

* function there\_is\_rule(rules, ps)

Return true or false

You can start with

* BC\_zookeeper\_student.html
1. **Something to think**
* What if there are more facts?
1. **Submission**
* Submit a document for 2) and the program for 5) by email.
	+ Due:
		- 11:59 pm, February 25, 2019 – with bonus 10%
		- 6:00 pm, February 27, 2019 – with the full marks
		- 6:00 pm, February 28, 2019 – with penalty 5%
		- 6:00 pm, March 1, 2019 – with penalty 10%
* Total marks: 12 (2 + 10)