Knowledge representation

Tutorial 4

18 October 2013

Correction of proposed exercises

1. Define a predicate `occur(+Ls, -Zs)` that succeeds if the list `Zs` is the list of the occurrence of `Ls`'s elements.

```
?- occur([a, b, a, a, b, c, a], X).

    X = [[a|4], [b|2], [c|1]] ;
    false.
```

2. Suppose that we have a set of denominations (coins of 1 euro, 2, banknotes of 5, 10, 20, 50, 100, 200, 500) and we want to know the number of possible ways to pay a certain amount. Define a predicate to compute this number.

Full Binary Trees

We will consider full binary trees where only the leaves have labels and where every node has exactly (strictly) 0 or 2 children.
We will represent the leaves by their label and the inner nodes by a dotted pair `[L|R]` where `L` and `R` denote the left and right subtree.

3. Define a predicate `is_binTree(+Tr)` that succeeds if `Tr` is a Prolog term representing a full binary tree.

4. Define a predicate `count_leaves(+Tr,-N)`, where `N` is a natural number and `Tr` is a full binary tree, that succeeds if `Tr` has exactly `N` leaves.

5. Define a predicate `depth_tree(+Tr,-N)`, where `N` is a natural number and `Tr` is a full binary tree, that succeeds if the tree `Tr` has a depth equal to `N`.
The depth (or height) of a tree is the length of the path from the root to the deepest node in the tree. A (rooted) tree with only one node (the root) has a depth of zero.

---

6. Define a predicate `explore(+Tr,-Ls)`, where `Tr` is a full binary tree labeled by natural numbers and `Ls` is a list, that succeeds if `Ls` is the list of leaves's labels encountered during depth-first traversal "right- left" of the tree. In addition, each label which is an odd number is replaced by the first bigger even number.

---

7. Define a predicate `same_frame(+Tr1,+Tr2)` that succeeds if the full binary trees `Tr1` and `Tr2` have the same set of leaves (at first, with the same number of occurrences, then, without this constraint).

---

8. Define a predicate `simplify(+Tr1,-Tr2)`, where `Tr1` and `Tr2` are full binary trees labeled by natural numbers, that succeeds if `Tr2` is the simplified version of `Tr1`. To simplify a tree, each node with two children leaves that have the same label is replaced by a leave with this label. At first, we will simplify only at one level of the tree, then, we will simplify recursively as long as the labels are equals.

Truth table

---

9. Define a predicate `table(+Vs, +E)` that writes the truth table of the expression `E` and where `Vs` is the list of variables of `E`.

Define the operators :

`~/1`, `^/2`, `v/2`, `=>/2`, `<=/2`, `<=>/2` and `<~>/2`

which are logical operators

`not`, `and`, `or`, `implication`, `inv implication`, `equivalence` and `xor`

respectively.

```
?- table([P,Q,R], (P => (Q => R)) => ((P => Q) => (P => R)) ).

  F  F  F  |  T
  F  F  T  |  T
  F  T  F  |  T
  F  T  T  |  T
  T  F  F  |  T
  T  F  T  |  T
  T  T  F  |  T
  T  T  T  |  T
true.
```

Proposed exercise

---

10. Define a predicate `countdown(+Ns,+K,-Lo)` that succeeds if `Ns` is a list of natural numbers from where we can compute the number `K` with the list of arithmetic operations specified by `Lo`.

```
?- countdown([4, 75, 10, 7, 25, 1], 405, Lo).

    Lo = [[7, +, 25, =, 32], [32, +, 1, =, 33],
          [33, *, 10, =, 330], [330, +, 75, =, 405]]
```