CS550-01-Spring2018 (Artificial Intelligence): questions about backtracking

1 message

Marie Roch - marie.roch@sdsu.edu <do-not-reply@blackboard.com> Sun, Apr 15, 2018 at 10:59 AM
Reply-To: "Marie Roch - marie.roch@sdsu.edu" <marie.roch@sdsu.edu>

Dear Students,

Some questions were asked about backtracking this weekend that I thought many of you might want to see, if you have already solved the problem, you need not read.

> Professor,
> I am having some trouble implementing and understanding the pseudo-code in the book/your slides compared to what was given in the code in the assignment for backtracking. I was hoping you could answer a few quick questions.
> 1. The pseudo-code says "if value is consistent with assignment" does this simply mean if the value isn't in the assignment set already?

No, it means if I were to bind the variable to this value, would there be any conflicts? The csp class has method nconflicts that will test for conflicts with a hypothetical value.

> 2. What are removals with respect to using the Inference in backtracking? One of the backtrack_util inferences uses AC3 with it, but there wasn't any comments in the code describing what to do with that variable in AC3, same as queue so I did not implement them originally.

AC3 should populate the queue if it is empty. When called from the mac function, mac populates the queue to only look at the neighbors of the variable that we are assigning. This of course can propagate as we go through the algorithm as AC3 appends to the queue when revisions are made.

As for removals, these are only used in the backtracking search. When backtracking makes an assignment, it needs to keep track of the value it has removed. While this could be done in your code, the csp class has method suppose that will return a removal list (should be single variable/values in this case, if we were doing more sophisticated search (e.g. conflict handling) it might be different.
The csp class's restore will put things back the way they were if things don't work out.

Here's an extended version of the headers in constraint_prop.py with additional comments that you may find helpful:

```
def AC3(csp, queue=None, removals=None):
    """AC3 constraint propagation

    csp - constraint satisfaction problem
    queue - list of constraints (might be None in which case they are
        populated from csp's variable list (len m) and neighbors (len k1...km):
        [(v1, n1), (v1, n2), ..., (v1, nk1), (v2, n1), (v2, n3), ... (v2, nk2),
         (vm, n1), (vk, n2), ..., (vk, nkm) ]
    removals - List of variables and values that have been pruned.  This is only
        useful for backtracking search which will enable us to restore things
        to a former point

    returns
        True - All constraints have been propagated and hold
        False - A variables domain has been reduced to the empty set through
            constraint propagation.  The problem cannot be solved from the
            current configuration of the csp.
    """

    # Hints:
    # Remember that:
    #    csp.variables is a list of variables
```

Queue: None - generate pairs
!None - used for backtrack to see if mac will propagate a queue s.t. it'll only be a subset of all variables

```
    #    cps.neighbors[x] is the neighbors of variable x

def revise(csp, Xi, Xj, removals):
    """Return true if we remove a value.
    Given a pair of variables Xi, Xj, check for each value i in Xi's domain
    if there is some value j in Xj's domain that does not violate the
    constraints.

    csp - constraint satisfaction problem
    Xi, Xj - Variable pair to check
    removals - list of removed (variable, value) pairs.  When value i is
        pruned from Xi, the constraint satisfaction problem needs to know
        about it and possibly updated the removed list (if we are maintaining
        one)
    """

    # Hint:Use csp's prune() method to manage the removals list
```

3. What are the return values for AC3 so that they correspond to how the inference "mac" would use it? Is it false if one of the domains gets restricted to an empty set and true otherwise?

Correct.

All my best,

Professor Roch