CS3243 **AY20/21 Sem1**

Lecture 7: October 7

Lecturer: MEEL, Kuldeep S. *Scribes: Song Qifeng*

7.1 Infer in CSP

7.1.1 Basic idea

In regular state-space search as we have seen before, the algorithm can do only one thing: search. In CSP, however, an algorithm can both search and do a specific type of inference called **constraint propagation**.

Constraint propagation means using the constraints to reduce the number of legal values for a variable (add to inference), and the newly added inference can in turn reduce the legal values for other variables, and the process continues.

7.1.2 Representation for inferences

Inference: $x \notin S$ means x does not take any value in set S.

e.g. $x \notin \{3\}$ means $x \neq 3$.

Method: $ComputeDomain(var, assign, inference)$ returns the set of values of var that var can take under current assignment and inference.

e.g. assign = [], $inference = x \notin \{2,3\}, domain = \{1,2,3,4\}$, $ComputeDomain(x, assign, inference)$ returns $x = \{1,4\}$.

Rationale: x can take any value except those in the inference.

e.g. assign = $[\{x = 1\}]$, $inference = x \notin \{2,3\}, domain = \{1,2,3,4\}$, $ComputeDomain(x, assign, inference)$ returns $x = \{1\}$.

Rationale: Since x has already been assigned a value, then the method will just return this value.

e.g. assign = [], $inference = x \notin \{1,2,3,4\}, domain = \{1,2,3,4\}$, $ComputeDomain(x, assign, inference)$ returns *failure*.

Rationale: There is not a value for x to take s.t. the constraint is not violated.

7.1.3 Algorithm of infer

Infer($prob, var, assign$):

```
$varQueue$ ←[var]
while $varQueue$ is not empty
    $y$ ←varQueue.pop()
    for each constraint $c$ where $y \in Vars(c)$
        for each $x \in Vars(c)$ besides $y$
            $S$ ←computeDomain(x,assign,infer)
            for each $v$ in $S$
                if no valid value exists for the rest of the variables (i.e. $Vars(c) - x$) s.t. $c[x = v]$ is satisfied
                    then $infer.add(x \notin v)$
            $T$ ←ComputeDomain(x,assign,infer)
            if $T = \{\}$
                then return $failure$
            if $T \neq S$
                then $varQueue.add(x)$
return $infer$
```

7.1.4 Variants of inference

$Infer$ can be expensive, and here are two ways to reduce the time of infer. However, the trade-off is, as we make the algorithm less expensive, it can also be less informative.

1. Forward checking: Do not add anything to varQueue.

2. Replace "if $(S \neq T)$" condition with "if $(size(T) = 1)$." This approach may seem very arbitrary and so it is, there is no deep reason for why we choose 1 as the threshold, in theory, we can pick any number as the threshold, however, 1 as the threshold works quite well in practice.

7.1.5 Storage of constraints

It is intuitive to use truth table to store constraints, however, as the size of the problem grows, the table can be too large.

In some cases, constraints can be written in math form, e.g. the noAttack constraint in N-queen problem can be written as $|x_i - x_{i+1}| > 1$.

7.1.6 Pick unassigned variables

We can pick the variable with the fewest valid values because it is more likely to fail. This is called **Minimum remaining value selection**.

7.1.7 Order domain value

We assign the value in the variable domain that is most likely to succeed.