

Constraint Programming: Search

- 1 Depth First Search
- 2 Backtracking
- 3 Forward Checking
- 4 Looking Ahead
- 5 Summary and Exercises

Search & Consistency

Constraint satisfaction through **search** of the solution space

- constraints used passively as a test
- assign values and try what happens
- examples: **backtracking**, **generate & test** (trivial)
- complete methods (either solution is found or inconsistency is proved)
- too slow (exponential): search through "clearly" bad assignments

Consistency/propagation techniques

- allow to remove inconsistent values from domains
- incomplete methods (some inconsistent values still in domains)
- relatively fast (polynomial)

Combination of both methods used

- subsequent assignment of values to variables
- after assignment, inconsistent values removed by consistency techniques

Depth First Search (DFS)

Depth first search of the solution space:
base search algorithm for CSPs

Two phases of the search

- **forward phase:** variables subsequently selected, partial assignment extended by assignment of consistent value (if exists) to another variable
 - after value selection, consistency tests are processed
- **backward phase:** if there is no consistent value for current variable, algorithm backtracks to earlier assigned value

Types of variables

- **past:** already selected variable (have assigned value)
- **current:** currently selected variable to be assigned a value
- **future:** variables which will be selected in the future

Core search procedure: DFS

Variables numbered for simplicity, assignment processed in given order

Initial call: labeling($G,1$)

procedure labeling(G,a)

if $a > |\text{edges}(G)|$ then return nodes(G)

for $\forall x \in D_a$ do

 if **consistent**(G,a) then % consistent(G,a) replaced by FC(G,a), LA(G,a), ...

$R := \text{labeling}(G,a + 1)$

 if $R \neq \text{fail}$ then return R

return fail

end labeling

R : assignment of variables or fail

Procedures consistent(G,i) will be described for binary constraints only

Backtracking (BT)

Backtracking verifies consistency of constraints leading from past variables to current variable at each step

Backtracking maintains consistency of constraints

- on all past variables
- on past and current variable

```
procedure BT( $G, a$ )
```

```
Q :=  $\{(V_i, V_a) \in \text{edges}(G), i < a\}$            % edges from past to current variable
```

```
Consistent := true
```

```
while non empty Q  $\wedge$  Consistent do
```

```
    choose and remove any edge  $(V_k, V_m)$  from Q
```

```
    Consistent := not revise( $V_k, V_m$ ) % if value is removed, domain is empty!
```

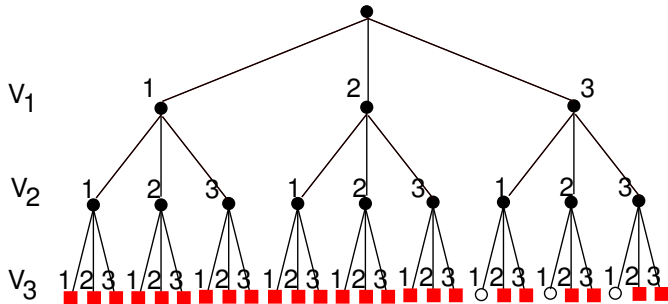
```
return Consistent
```

```
end BT
```

Example: Backtracking

Constraints: V_1, V_2, V_3 in $1 \dots 3$, $V_1 \neq 3 \times V_3$

Solution space:



- red boxes: failed attempt for assignment, no solution
- empty circles: solution found
- black circles: inner node representing partial assignment

Forward Checking (FC)

FC is an extension of backtracking

In addition, FC maintains consistency between current and future variables

procedure FC(G, a)

$Q := \{(V_i, V_a) \in \text{edges}(G), i > a\}$

% addition of arcs from future to current variable

Consistent := true

while non empty $Q \wedge$ Consistent do

 choose and remove any edge (V_k, V_m) from Q

 if revise((V_k, V_m)) then

 Consistent := $(|D_k| > 0)$

% empty domain means fail

return Consistent

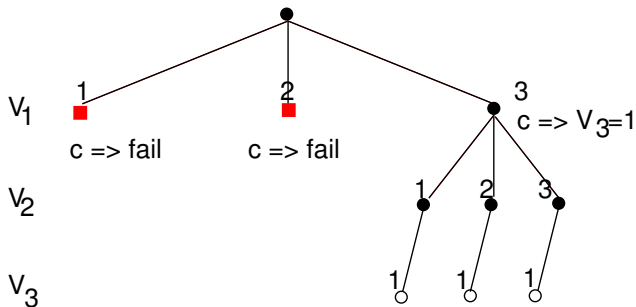
end FC

Edges from past to current variables is not necessary to test

Example: Forward Checking

Constraints: V_1, V_2, V_3 in $1 \dots 3$, $c : V_1 \# = 3 \times V_3$

Solution space:



Looking Ahead (LA)

LA is an extension of FC, LA maintains arc consistency

In addition, LA maintains consistency between all future variables

procedure LA(G, a)

$Q := \{(V_i, V_a) \in \text{edges}(G), i > a\}$

% start with edges leading to a

Consistent := true

while non empty $Q \wedge$ Consistent do

 choose and remove any edge (V_k, V_m) from Q

 if revise((V_k, V_m)) then

$Q := Q \cup \{(V_i, V_k) \mid (V_i, V_k) \in \text{edges}(G), i \neq k, i \neq m, i > a\}$

 Consistent := $(|D_k| > 0)$

return Consistent

end LA

- edges from past variables to current variable are not necessary to test again
- this LA procedure is based on AC-3, other AC algorithms can be also applied

LA maintains arc consistency: since LA(G, a) applies AC-3,

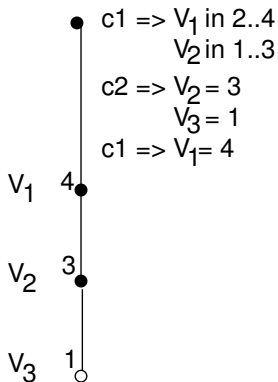
initial consistency must be computed by AC-3 before search starts

Example: Looking Ahead (with AC-3)

Constraints: V_1, V_2, V_3 in $1 \dots 4$, $c1 : V_1 \neq V_2$, $c2 : V_2 = 3 \times V_3$

Solution space:

- initial consistency is computed (by AC-3 algorithm) before search

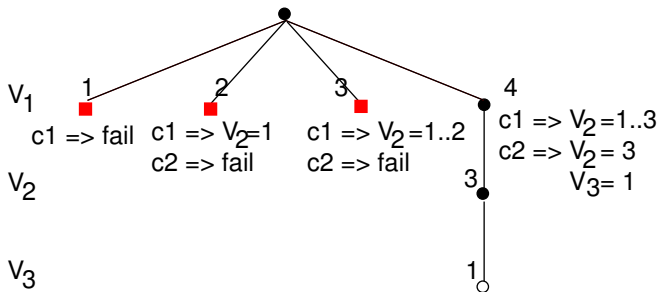


Example: Looking Ahead with AC-1

Constraints: V_1, V_2, V_3 in $1 \dots 4$, $c1 : V_1 \# > V_2$, $c2 : V_2 \# = 3 \times V_3$

Solution space (when AC-1 is used instead of AC-3):

- initial consistency before search is not computed
 - AC-1 algorithm repeats revisions of all arcs in cycles unless domains of all variables are stable (do not change)
- ⇒ AC-1 makes the problem arc consistent as soon as the value of current variable is assigned



Summary of Algorithms

Backtracking (BT) maintains at step a constraints

$$c(V_1, V_a), \dots, c(V_{a-1}, V_a)$$

from past variables to current variable

Forward checking (FC) maintains at step a constraints

$$c(V_{a+1}, V_a), \dots, c(V_n, V_a)$$

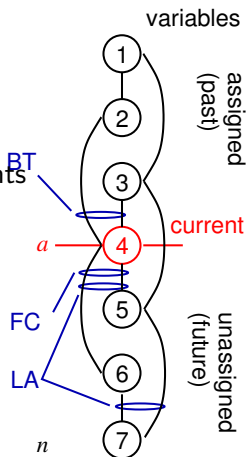
from future variables to current variable

Looking Ahead (LA) maintains at step a constraints

$$\forall l(a \leq l \leq n), \forall k(a \leq k \leq n), k \neq l : c(V_k, V_l)$$

from future variables to current variable

between future variables



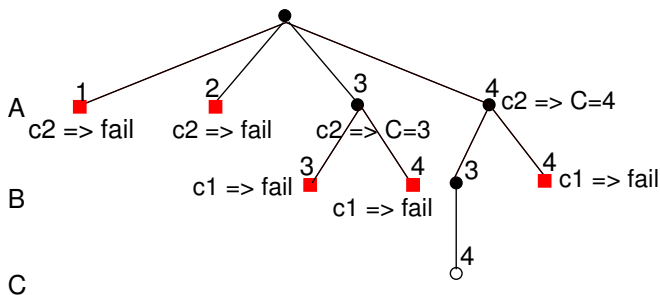
Exercise 1.

1. Write solution space for constraints

A in 1..4, B in 3..4, C in 3..4, c1: $B \neq C$, c2: $A \neq C$

when using forward checking and ordering of variables A,B,C. Explain what types of propagation happens in each node.

Solution:



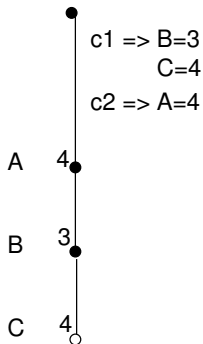
Exercise 2.

2. Write solution space for constraints

A in 1..4, B in 3..4, C in 3..4, c1: $B \neq C$, c2: $A \neq C$

when using looking ahead and ordering of variables A,B,C. Explain what types of propagation happens in each node.

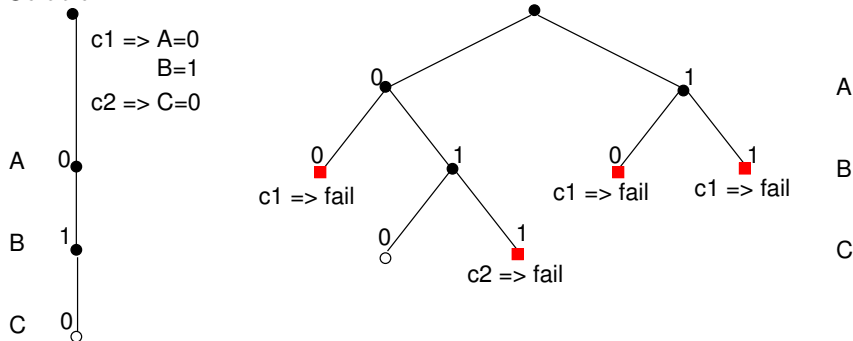
Solution: (initial propagation computed by AC-3)



Exercises 3. and 4.

3. Write and compare solution spaces for constraints domain($[A,B,C],0,1$), $c1: A \neq B-1$, $c2: C \neq A*B$ when using backtracking and looking ahead. Explain what types of propagation happens in each node.

Solution:



4. Present on some example differences between forward checking and looking ahead.