CS440/ECE448: Intro to Artificial Intelligence

Lecture 6: More on constraint satisfaction problems

Prof. Julia Hockenmaier
juliahmr@illinois.edu

http://cs.illinois.edu/fa11/cs440

Tuesday's key concepts

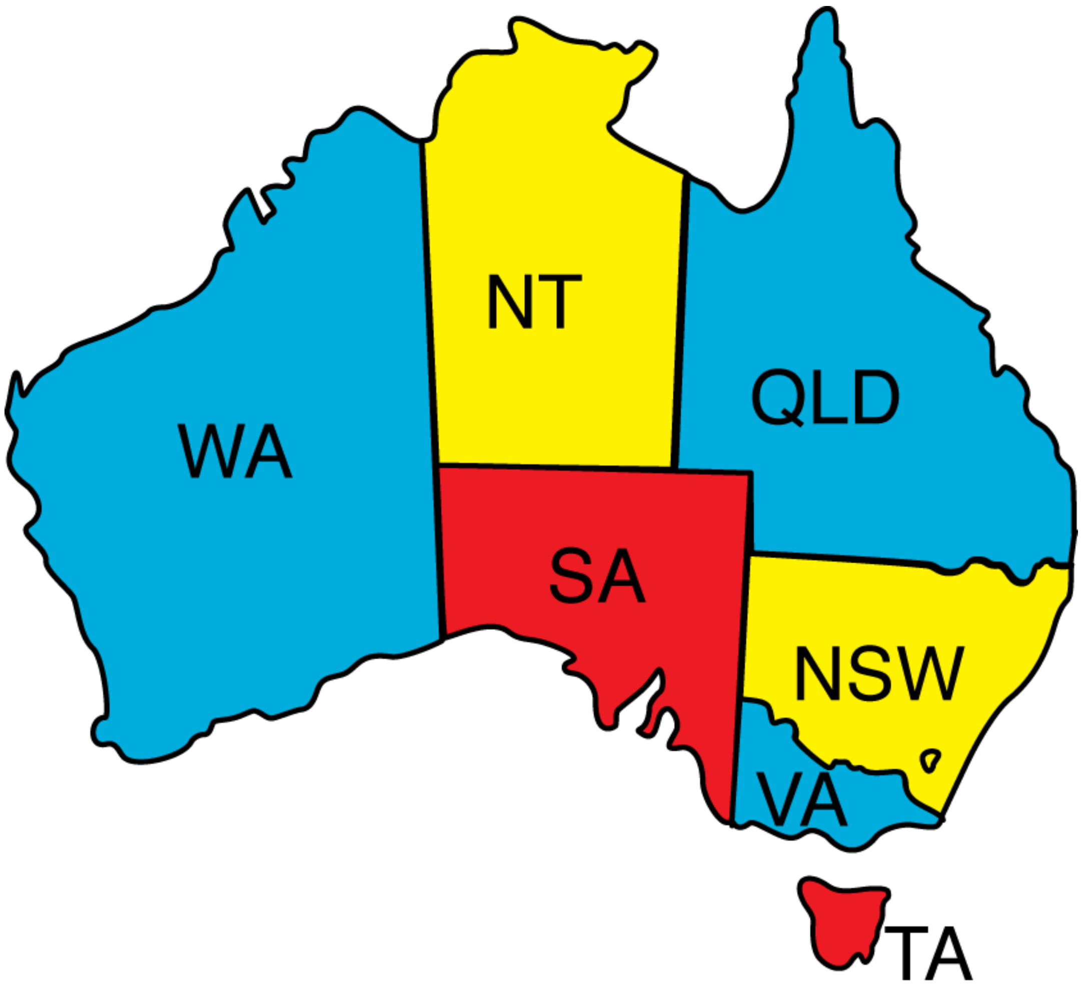
Constraint satisfaction problems:

Given: a set of *n* variables $X_1..X_n$, with domains (sets of possible values) $D_1 \ldots D_n$, and a set of *m* constraints $C_1 \ldots C_m$

Task: assign a value from D_i for each X_i subjects to the constraints.

CSP 1:
Map coloring
(Binary constraints)

Map coloring: a solution for N=3



Constraint satisfaction problems are defined by…

- **a set of variables** X:

  $\{WA, NT, QLD, NSW, VA, SA, TA\}$

- **a set of domains** D_i (possible values for variable x_i):

  $D_{WA} = \{red, blue, green\}$

- **a set of constraints** C:

  $\{\langle (WA,NT), WA \neq NT \rangle, \langle (WA,QLD), WA \neq QLD \rangle, \ldots\}$

  scope *relation*

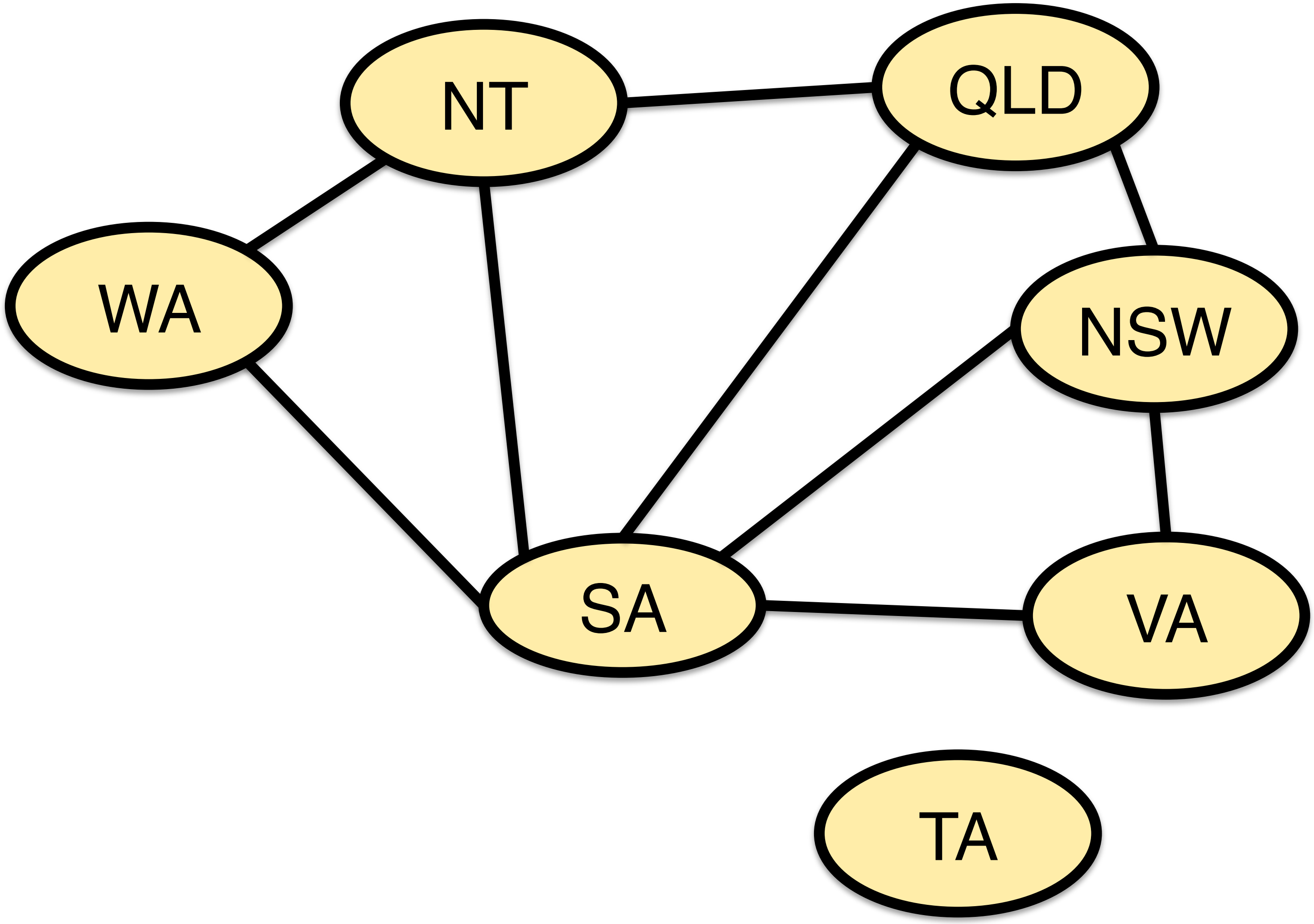
States and solutions

Each **state** is a complete or partial assignment of values to variables:

state35 = {WA=red, NT=blue, QLD= green, NSW= red, VA= green, SA= blue, TA= red};
state23 = {WA = red}

Legal assignments don't violate any constraints.
Solutions are complete legal assignments

Binary constraints: constraint graph



Consistency

Node consistency: X is node-consistent iff each element in D_X satisfies unary constraints on X

Arc consistency: X is arc-consistent iff for each C(X, Y) and for each $x \in D_X$ there is a $y \in D_Y$ such that the assignment {X=x, Y=y} satisfies C(X,Y).

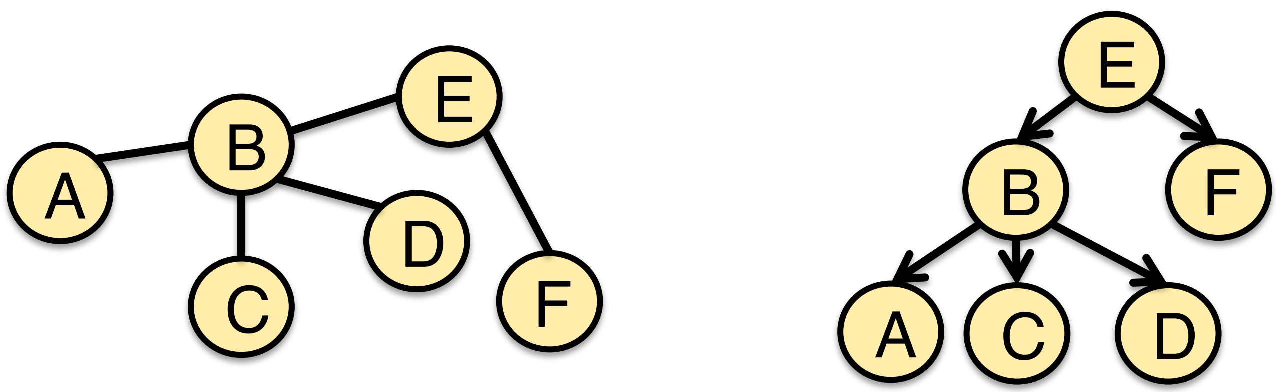
Path consistency: {X,Y} are path consistent wrt. Z iff for every $x \in D_X$ and $y \in D_X$ there is a $z \in D_Z$ such that the assignment {X=x,Y=y,Z=z} satisfies C(X,Z) and C(Y,Z)

AC-3

```
// Is the CSP c arc-consistent?
function AC3(CSP c)
   input: CSP c = (X,D,C)
   local: queue q ← all arcs C(X,Y) in c
   while q ≠ () do:
      // Can C(X,Y) be satisfied?
      (X,Y) = pop(q);
      // if domain(X) needs to be shrunk:
      if revise(c,X,Y):
         // Exit if CSP can't be solved:
         if domain(X) == () return false;
         // Are X's neighbors still okay?
         foreach Z in X.NEIGHBORS\{Y}:
            q ← push(q,(Z, X));
   return true;
```

Tree-structured constraint graphs

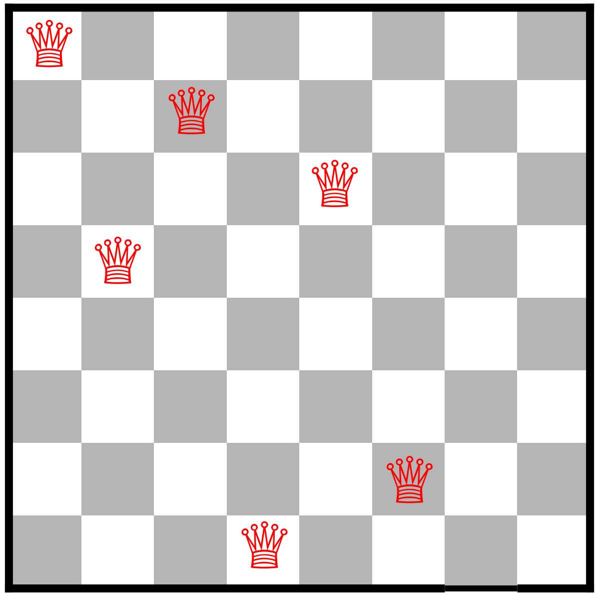
- Any two nodes connected by a single path
- With n vertices, there are n–1 edges
- Can be solved in linear time ($O(nd^2)$)



CSP 2:
8 queens puzzle

(Combining search and CSP inference)

The 8-queens puzzle

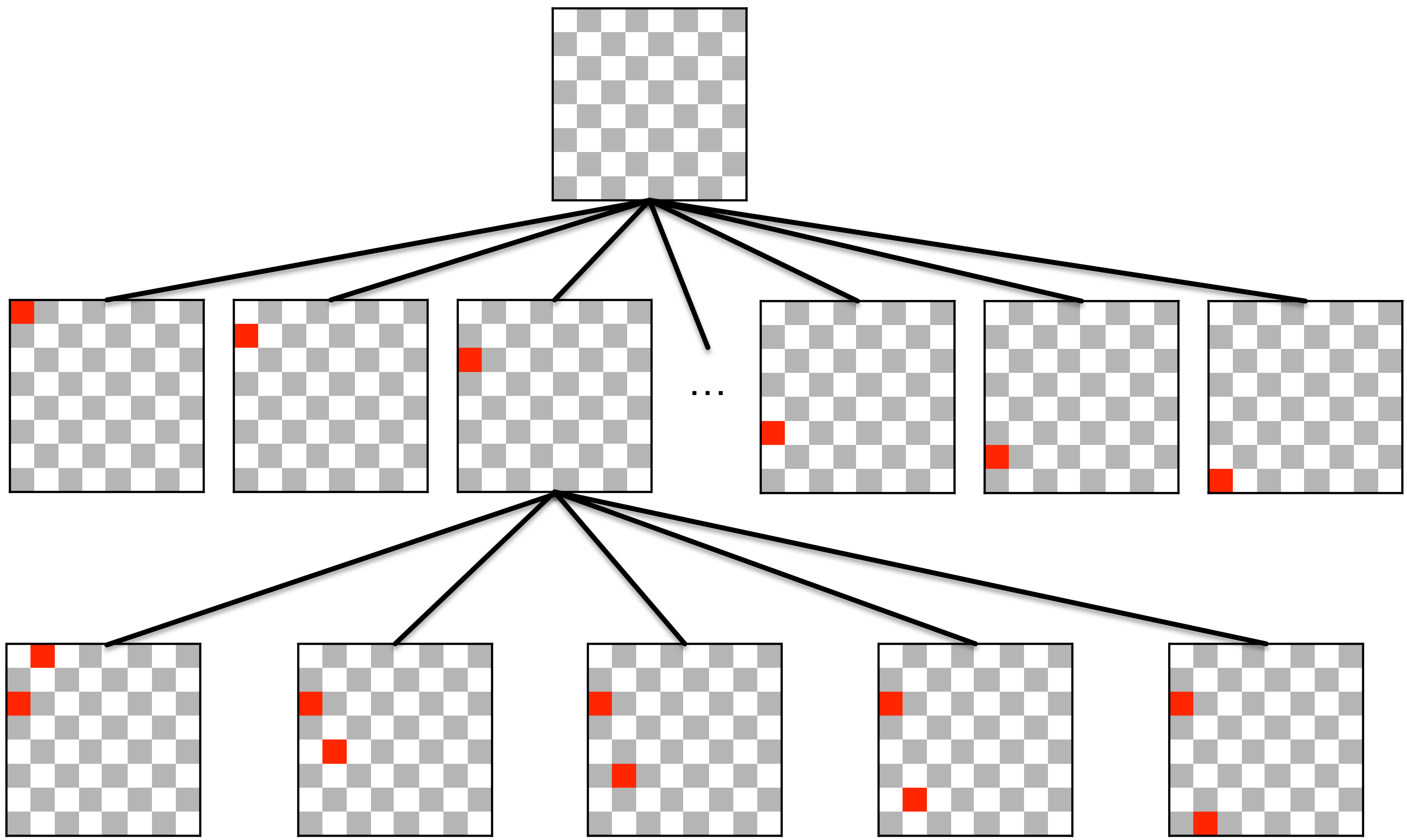


The 8-queens puzzle has multiple solutions

Cannot be solved by constraint propagation alone

[*underdetermined* CSP]

Search tree for 8-queens



Branching factor of 8-queens

The branching factor is at most 8:

– We can *order* the variables:
  (CSPs are **commutative**)
  1. place queen in column A,
  2. place queen in column B,
  3. ...

– The constraints restrict the domains of the remaining variables

Ordering the variables

D_X = {R,G,B} C(X,Y): X ≠ Y
D_Y = {R} C(X,Z): X ≠ Z
D_Z = {G,B}

Which variable should we consider first?

Minimum remaining values heuristic:

The variable with the *smallest domain*

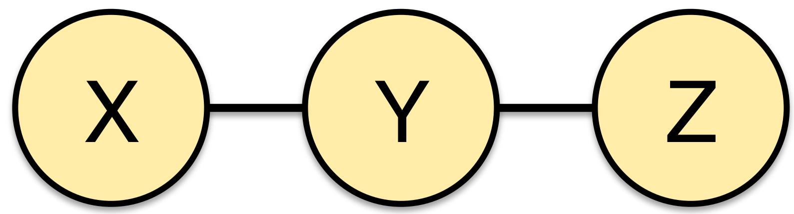
Degree heuristic:

The variable that has *the most constraints*

Caveat: implied constraints

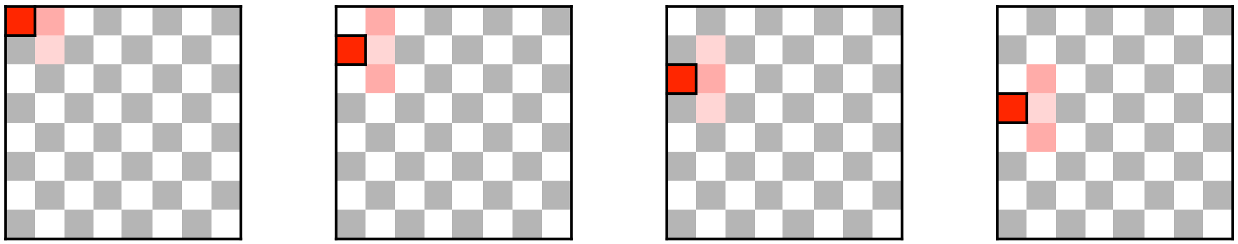
$D_X = D_Y = D_Z = \{0,1,2,3,4,5,6,7,8,9\}$

$C(X, Y): Y = X^2$
$C(Y, Z): Z = Y^2$



If we choose X as the root of the tree, we need to run DAC down the tree and up again.

Ordering the values



Which values of X should we consider first?

Least-constraining value heuristic:

Pick the *most likely* value first

(= the one which rules out the fewest choices for other variables)

Interleaving search and inference

Search: assign (guess) value x for variable X

Forward checking algorithm:
Check that X is arc-consistent with all remaining variables Y

Maintaining-Arc-Consistency algorithm:
Run AC3 on all C(Y,X) constraints to check overall arc-consistency

CSP 3:
Cryptarithmetic
(Global constraints)

Cryptarithmetic as CSP

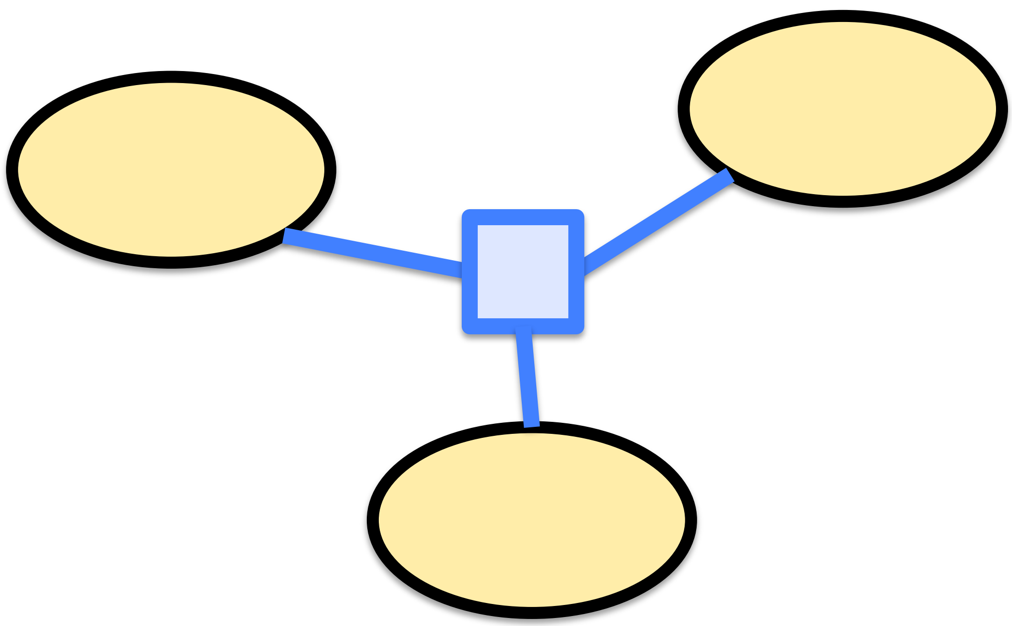
```
    TWO
+   TWO
-------
=  FOUR
```

Task: assign a digit to each letter

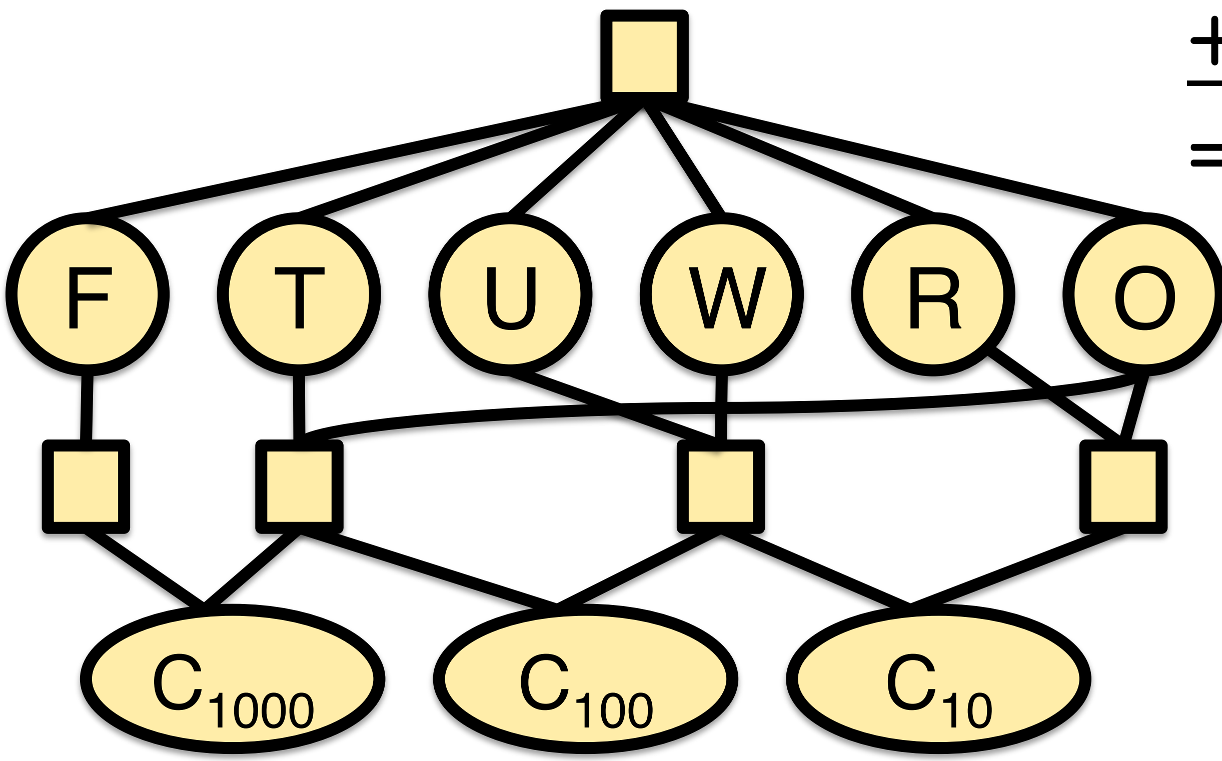
Constraints:
Each letter has a unique digit
(= AllDiff constraint)
The result has to be a valid sum

Hypergraphs

In a hypergraph, **hyperedges** connect multiple vertices:

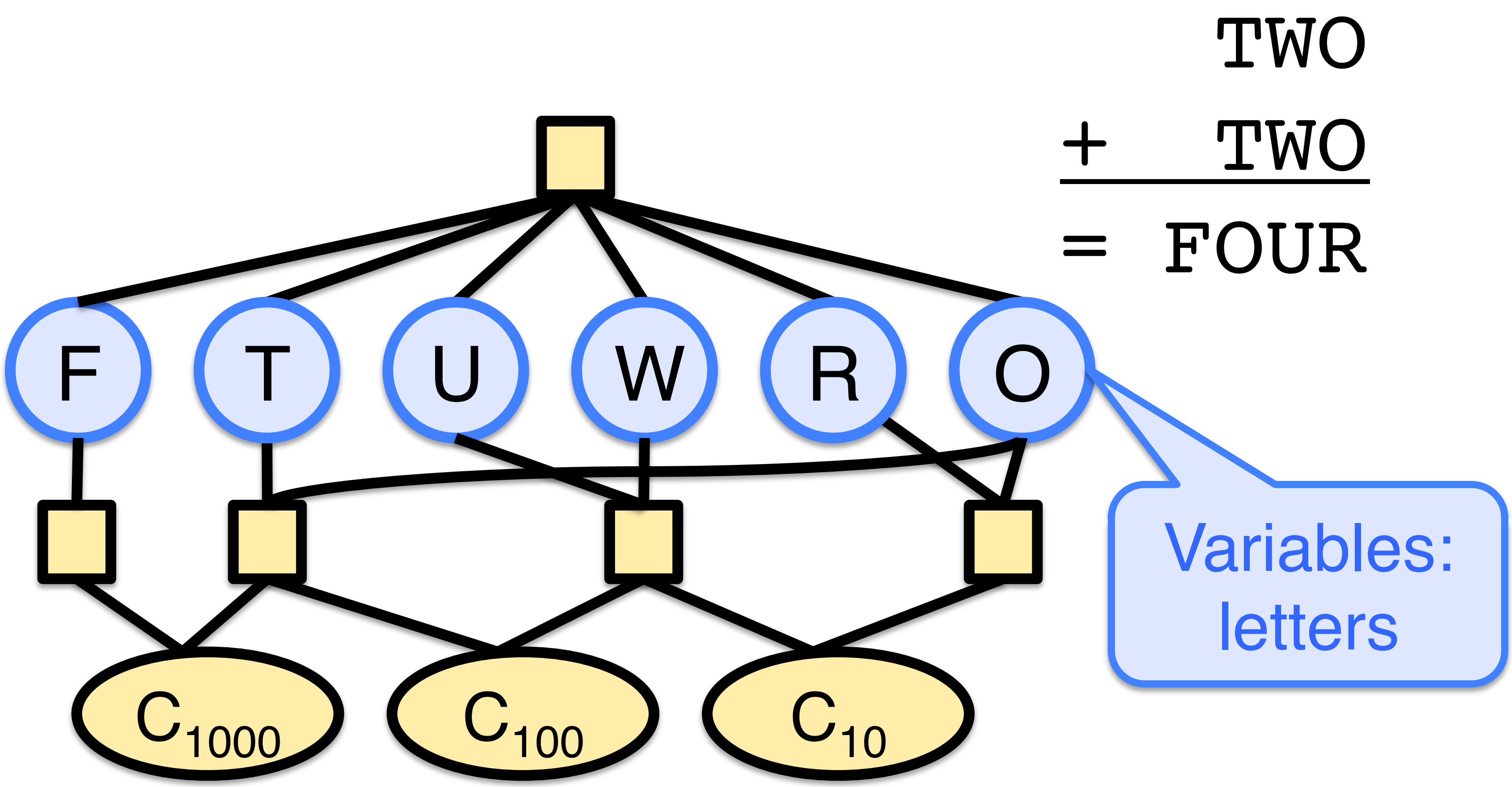


Global (n-ary) constraints: Constraint Hypergraph

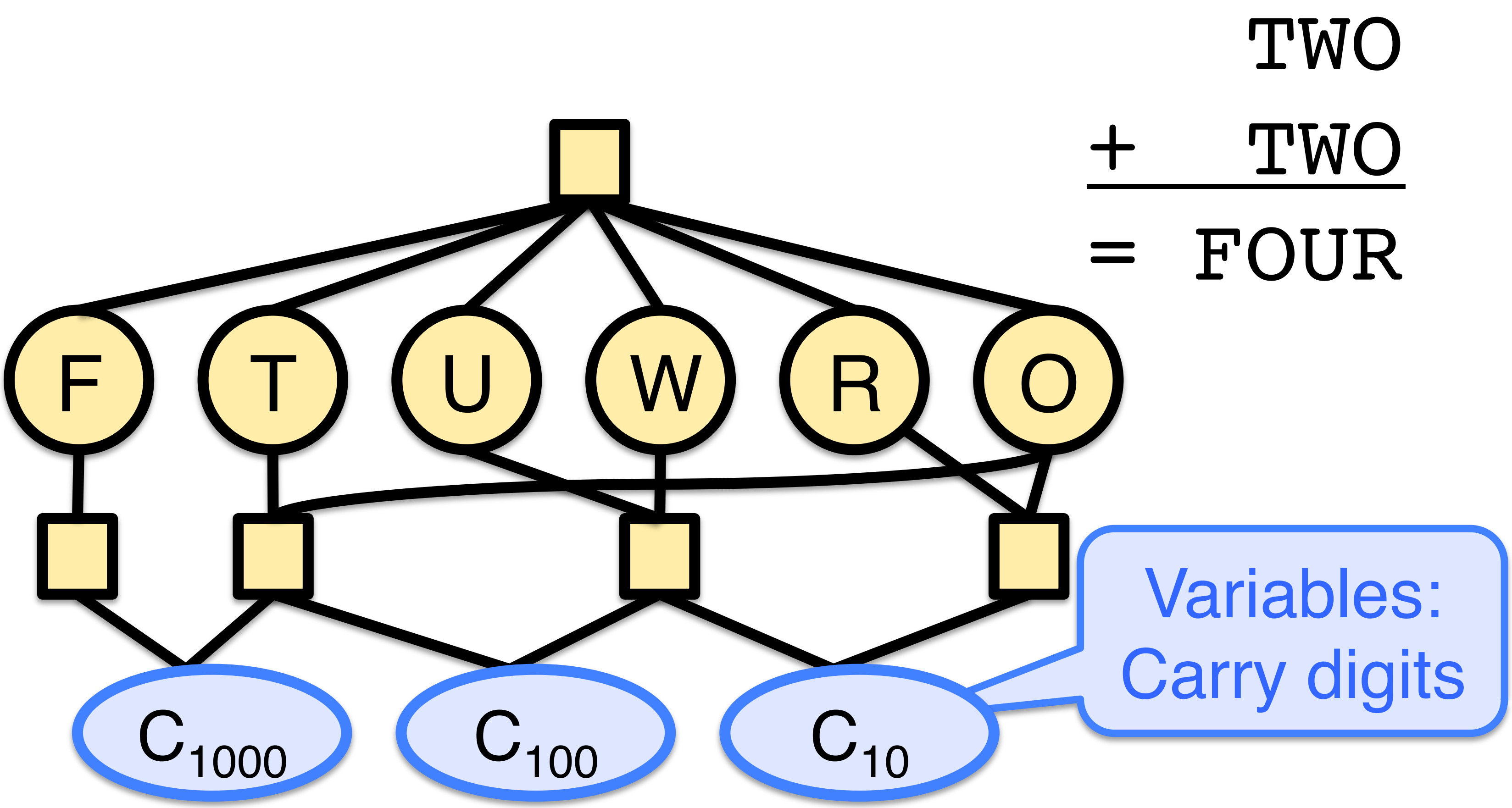



```
   TWO
+  TWO
= FOUR
```

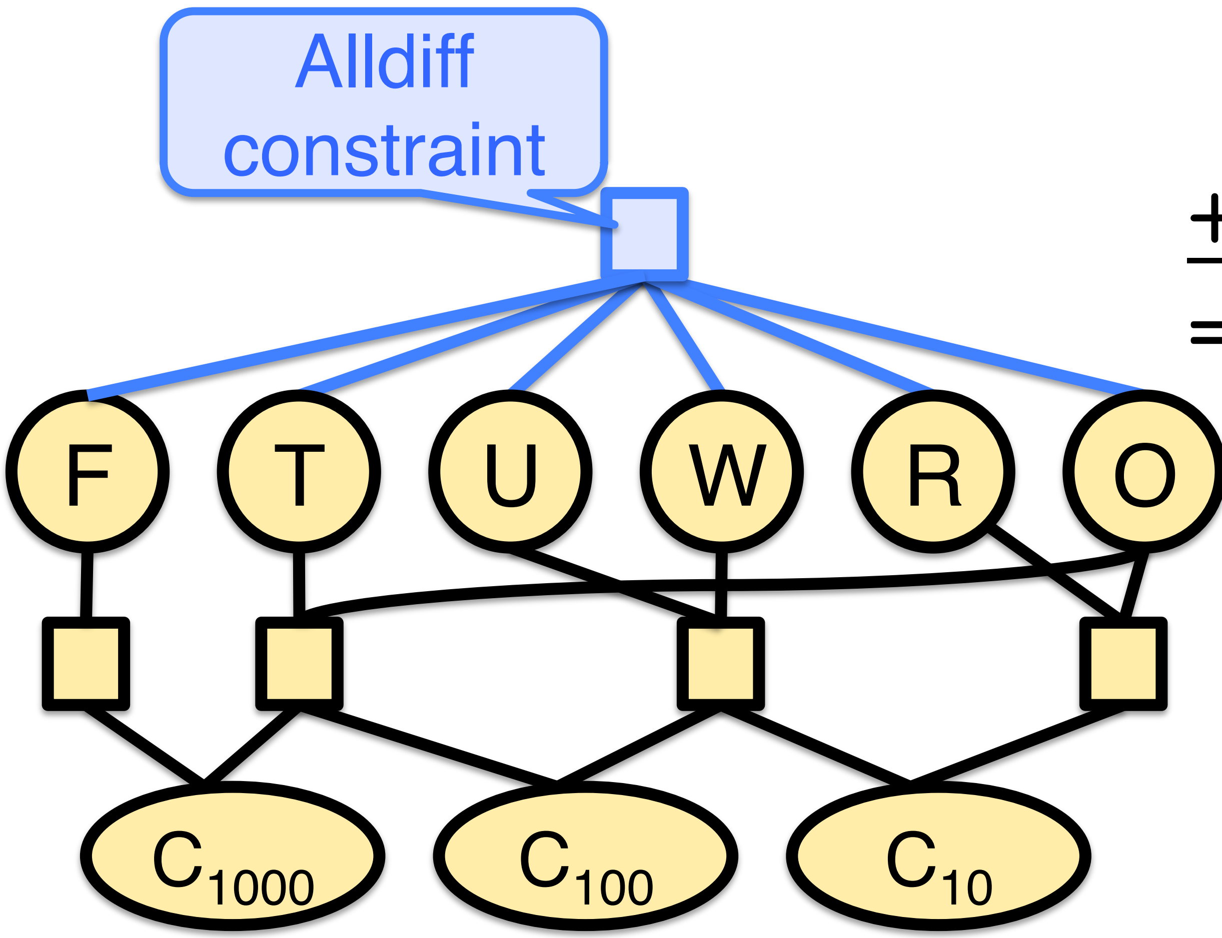
Global (n-ary) constraints: Constraint Hypergraph



Global (n-ary) constraints: Constraint Hypergraph

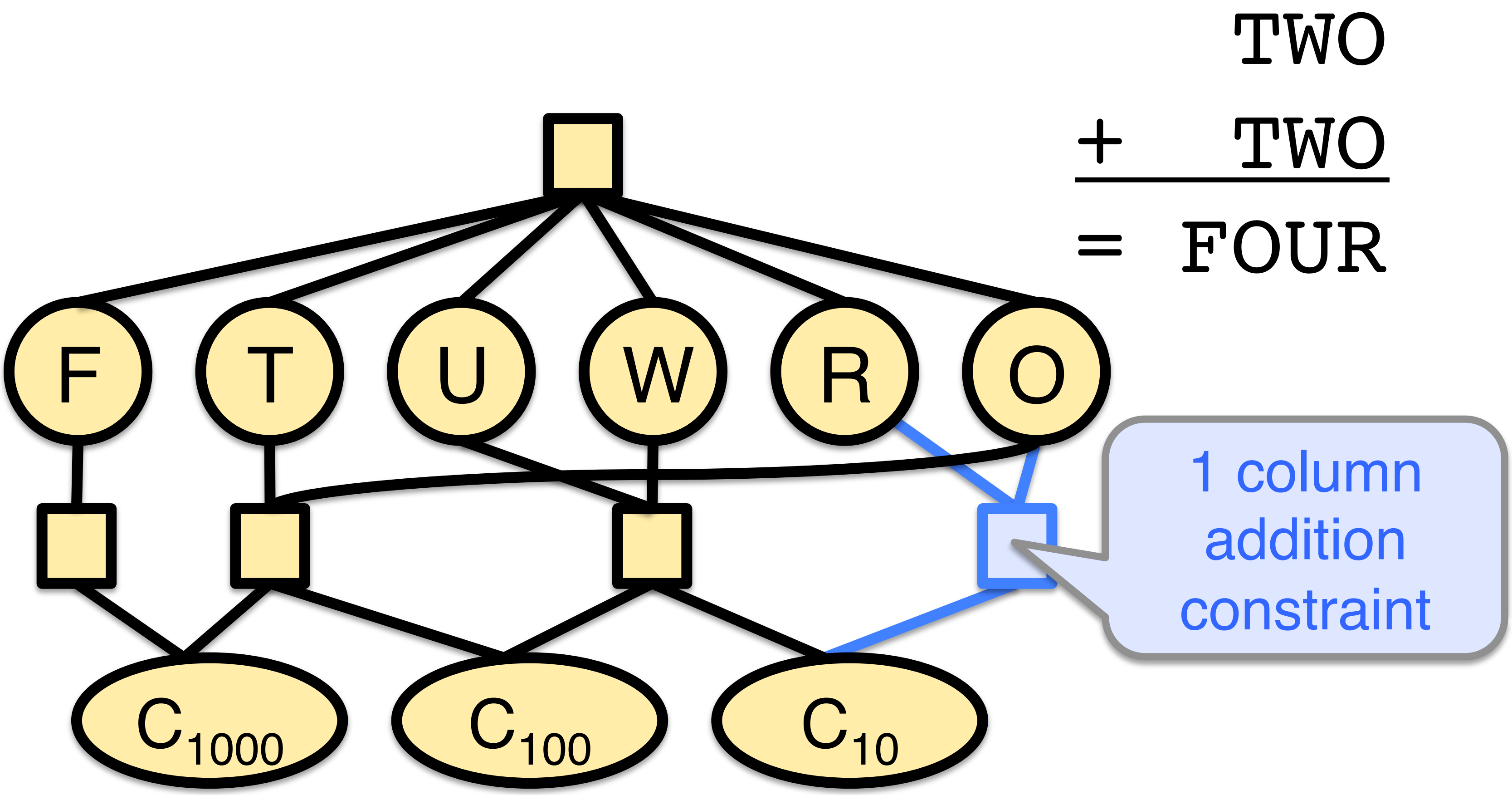


Global (n-ary) constraints: Constraint Hypergraph

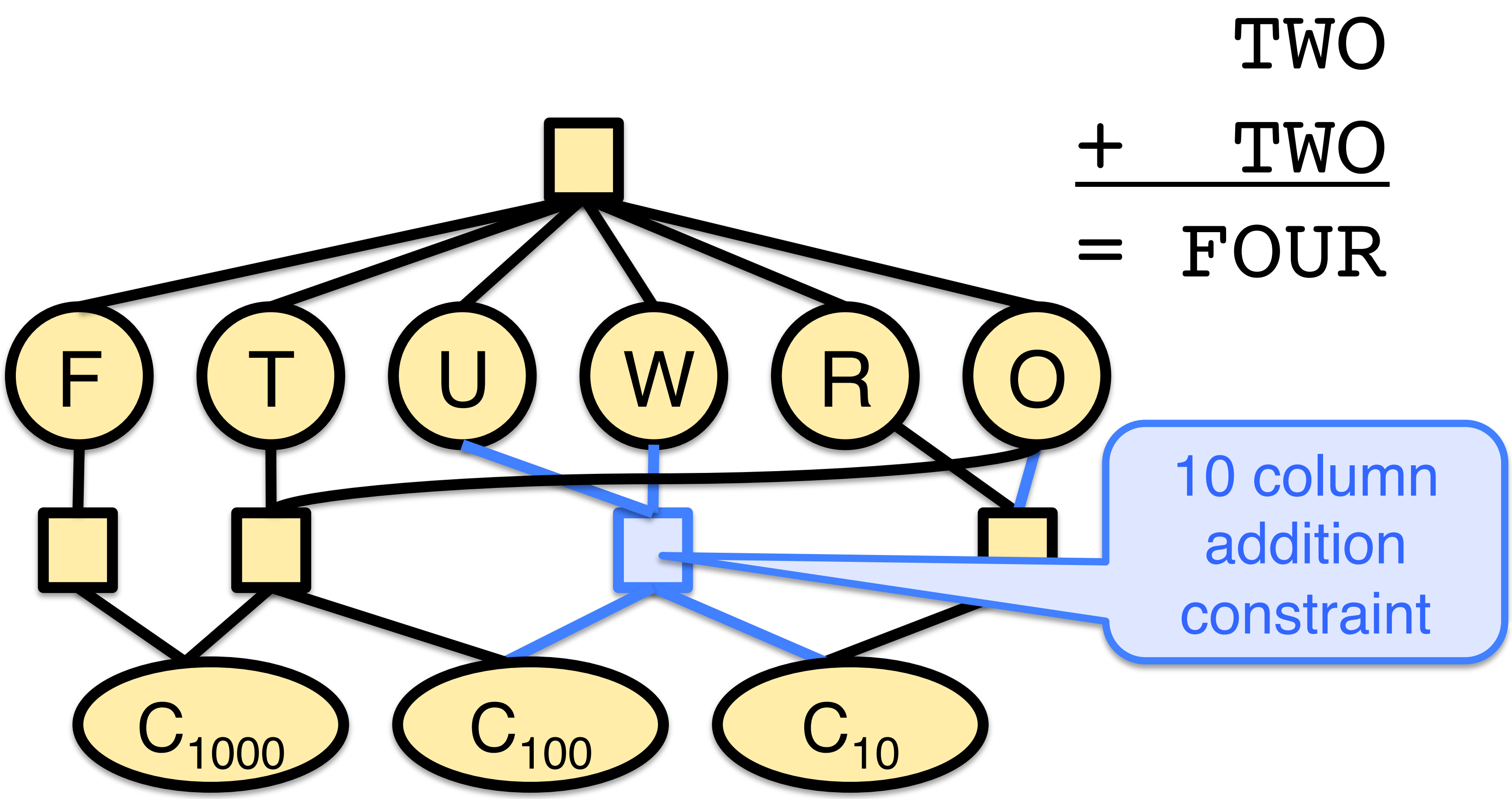



```
     TWO
+    TWO
--------
=   FOUR
```

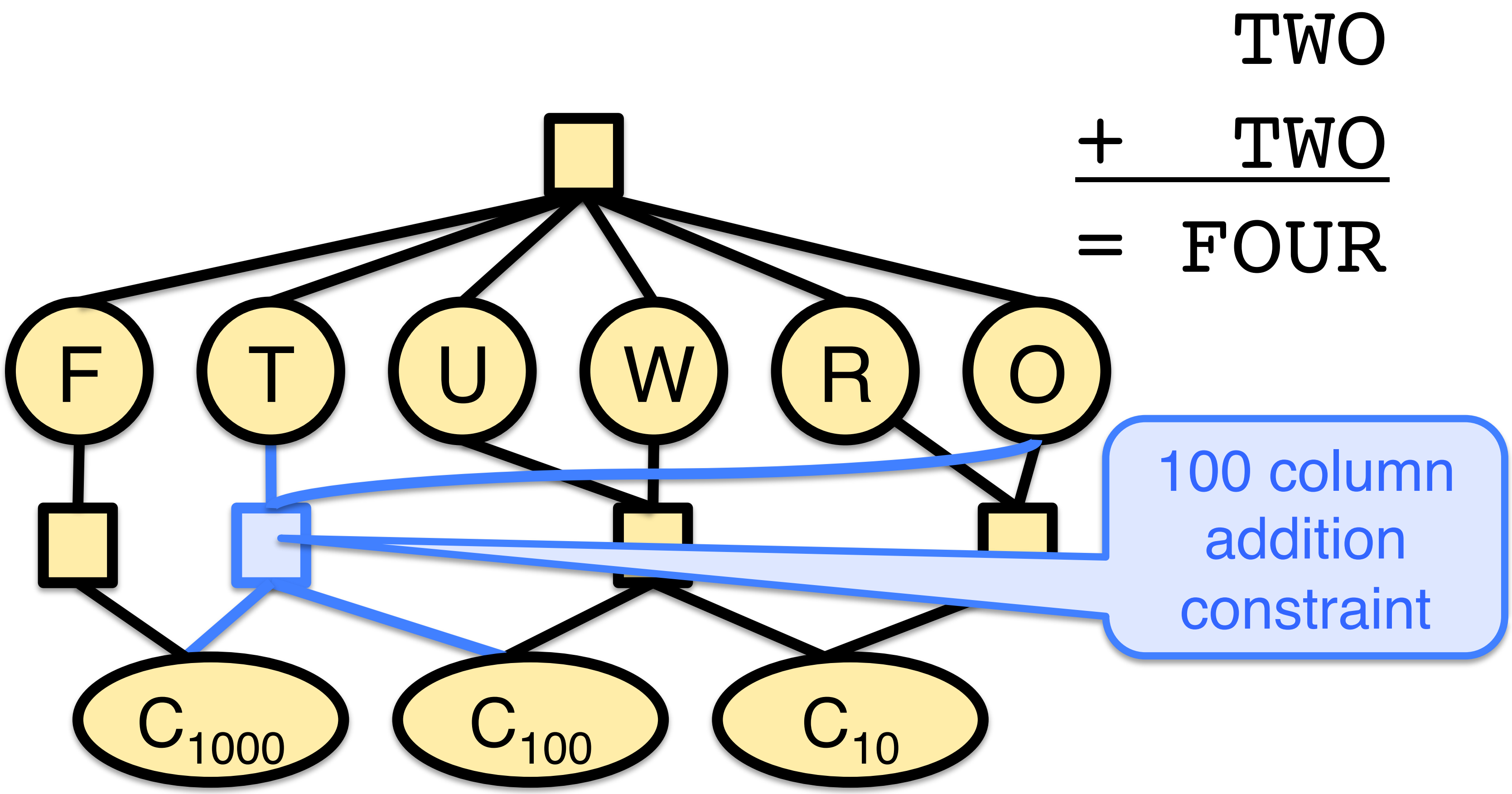
Global (n-ary) constraints: Constraint Hypergraph



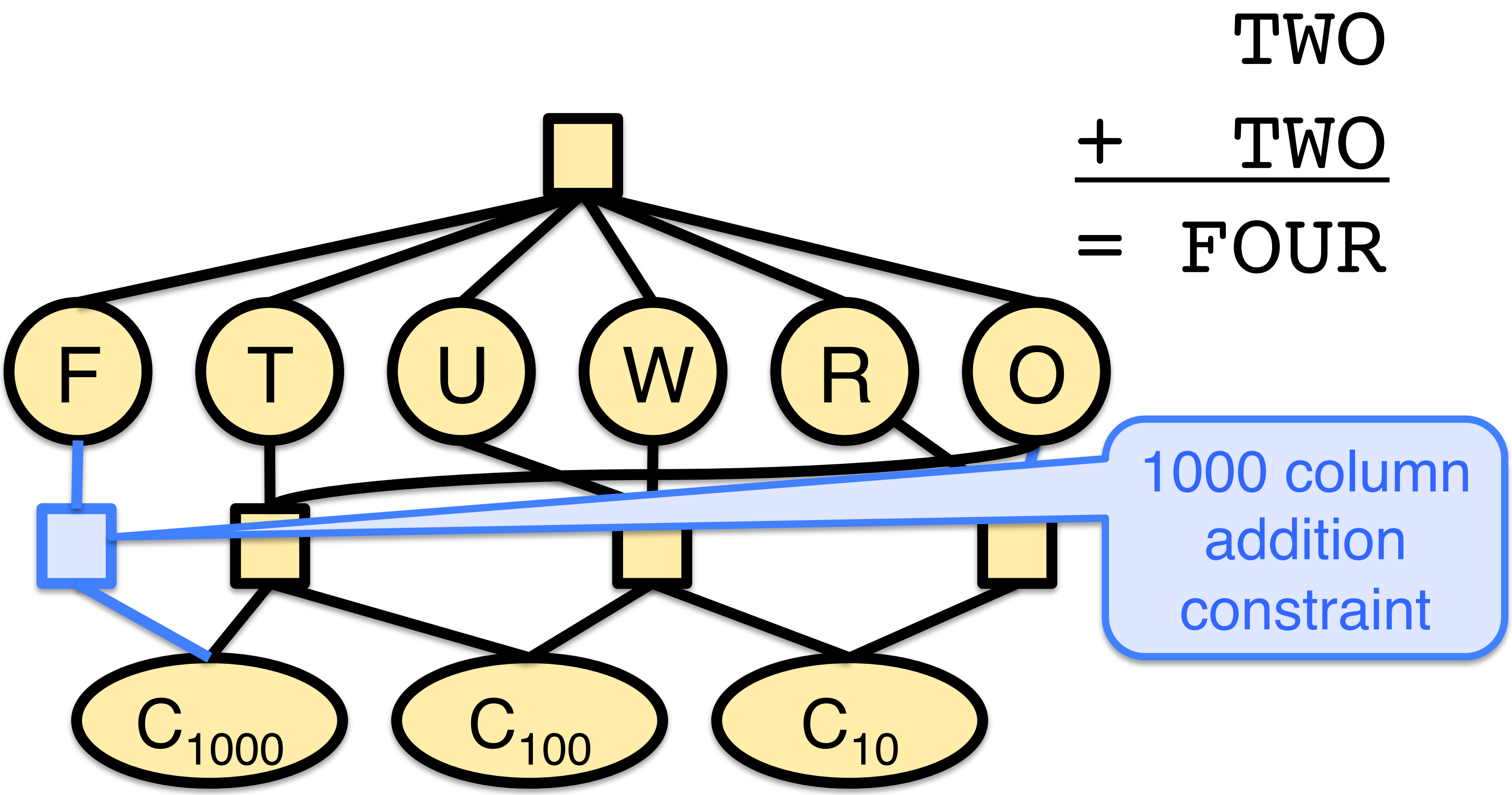
Global (n-ary) constraints: Constraint Hypergraph



Global (n-ary) constraints: Constraint Hypergraph



Global (n-ary) constraints: Constraint Hypergraph



Constraint propagation: Global constraints

Some n-ary constraints can be directly translated into a set of binary constraints:

Global constraint: AllDiff(X,Y,Z)
New binary constraints:

$C(X,Y): X \neq Y$
$C(X,Z): X \neq Z$
$C(Y,Z): Y \neq Z$

NB: Special purpose algorithms are often faster.

Global constraints

With additional auxiliary variables, any n-ary constraint can be translated into a set of binary constraints:

Ternary constraint:	C(X,Y,Z): $X+Y = Z$
Aux. variable A:	$d_D = \{\langle a_1,a_2 \rangle \mid a_1 \in d_X,\ a_2 \in d_Y\}$
New constraints:	C(A,Z): $a_1 + a_2 = Z$
	C(A,X): $a_1 = X$
	C(A,Y): $a_2 = Y$

NB: Special purpose algorithms are often faster

CSP 3:
Scheduling

(Continuous domains)

Job-shop scheduling

Task: schedule the steps required to assemble a car.

Constraints:

- Each step takes a certain amount of time
- Some steps need to happen before others
- Some steps require the same tools (can't happen at the same time)
- The car needs to be assembled by 5pm

Scheduling as CSP

Variables: *{WheelLF, WheelRF, … Engine,…}*
Domain: *8:00am…5:00pm*
Constraints:

- Front axle assembly takes 10 minutes, and has to happen before the front wheels:

  $AxleF + 10 \leq WheelLF$
  $AxleF + 10 \leq WheelRF$

- *Front and rear axle require the same tool:*

  $(AxleF + 10 \leq AxleR$ ***or*** $AxleR + 10 \leq AxleF)$

Bounds consistency: large (or continuous) domains

Continuous or large finite domains are represented by lower and upper bounds:
[lower… upper]

You want to invest $2000 in companies A and B.
A's shares cost $2, B's cost $1:
A: [0…1000] B: [0…2000]
You need to buy at least 100 shares of each:
Bounds propagation:
A: [100…950] B: [100..1800]

To conclude…

Today's key concepts

Combining CSP search and inference:
Ordering variables (minimum remaining value, degree heuristics)
Ordering values (forward checking, MAC)

Global constraints:
Constraint hypergraph; auxiliary variables
Continuous domains:
bounds consistency

Your tasks

Reading
Ch. 6.3, 6.5

Compass quiz:
up at 2pm