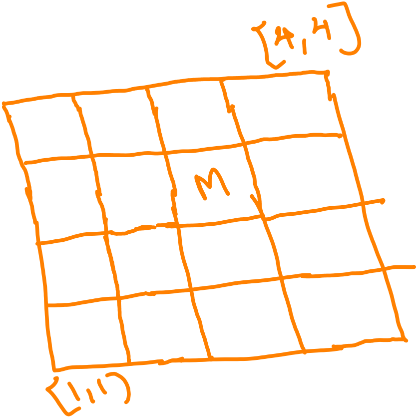
Questions regarding assignment2



Curiosity example

1. Query: Who killed the cat?
   KB $\models$ $Kills(x, Tuna)$?

Curiosity example

1. Query: Who killed the cat?

   KB $\models$ $Kills(x, Tuna)$?

- Nonconstructive proofs:

Curiosity example

1. Query: Who killed the cat?

   KB $\models$ $Kills(x, Tuna)$?

- Nonconstructive proofs:

  $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$, $\neg Kills(x, Tuna)$

Curiosity example

1. Query: Who killed the cat?

   KB $\models$ $Kills(x, Tuna)$?

- Nonconstructive proofs:

  $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$, $\neg Kills(x, Tuna)$

- Bind once and backtrack

Curiosity example

1. Query: Who killed the cat?

   KB $\models$ $Kills(x, Tuna)$?

   - Nonconstructive proofs:

     $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$, $\neg Kills(x, Tuna)$

   - Bind once and backtrack

2. The query in 1. is different from checking whether KB $\models \exists x\, Kills(x, Tuna)$?

Curiosity example

1. Query: Who killed the cat?

   KB $\models$ $Kills(x, Tuna)$?

- Nonconstructive proofs:

  $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$, $\neg Kills(x, Tuna)$

- Bind once and backtrack

2. The query in 1. is different from checking whether
   KB $\models \exists x \, Kills(x, Tuna)$?

   $\neg \exists x \, Kills(x, Tuna) \equiv \forall x \, \neg Kills(x, Tuna)$

Curiosity example

1. Query: Who killed the cat?

   KB $\models$ $Kills(x, Tuna)$?

   - Nonconstructive proofs:

     $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$, $\neg Kills(x, Tuna)$

   - Bind once and backtrack

2. The query in 1. is different from checking whether KB $\models \exists x\, Kills(x, Tuna)$?

   $\neg \exists x\, Kills(x, Tuna) \equiv \forall x\, \neg Kills(x, Tuna)$

   - Different substitutions of x is allowed:

Curiosity example

1. Query: Who killed the cat?

   KB $\models \underbrace{Kills(x, Tuna)}$?

- Nonconstructive proofs:

  $Kills(Jack, Tuna) \lor Kills(Curiosity, Tuna)$, $\underbrace{\neg Kills(x, Tuna)}$

- Bind once and backtrack

2. The query in 1. is different from checking whether

   KB $\models \underbrace{\exists x\, Kills(x, Tuna)}$?

   $\neg\exists x\, Kills(x, Tuna) \equiv \forall x\, \neg Kills(x, Tuna)$

- Different substitutions of x is allowed:

  $Kills(Jack, Tuna) \lor \underbrace{Kills(Curiosity, Tuna)}$, $\underbrace{\neg Kills(x, Tuna)}$

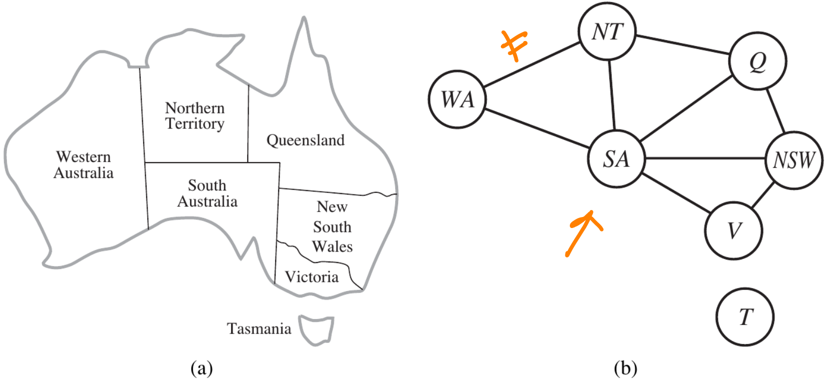


Figure 6.1 (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem (CSP). The goal is to assign colors to each region so that no neighboring regions have the same color. (b) The map-coloring problem represented as a constraint graph.

Constraint Satisfaction Problem

- Australian map coloring problem

Constraint Satisfaction Problem

- Australian map coloring problem
- Problem definition : X, D and C

$\{Red, G, B\}$

Constraint Satisfaction Problem

- Australian map coloring problem
- Problem definition : X, D and C
  1. $X = \{X_1, X_2, \ldots, X_n\}$

Constraint Satisfaction Problem

- Australian map coloring problem
- Problem definition : X, D and C
  1. $X = \{X_1, X_2, \ldots, X_n\}$
  2. $D = \{D_1, D_2, \ldots, D_n\}$

Constraint Satisfaction Problem

- Australian map coloring problem
- Problem definition : X, D and C
  1. $X = \{X_1, X_2, \ldots, X_n\}$
  2. $D = \{D_1, D_2, \ldots, D_n\}$
  3. C is a set of constraints.

Constraint Satisfaction Problem

- Australian map coloring problem
- Problem definition : X, D and C
  1. $X = \{X_1, X_2, \ldots, X_n\}$
  2. $D = \{D_1, D_2, \ldots, D_n\}$
  3. C is a set of constraints.
- Consistent assignment

Constraint Satisfaction Problem

- Australian map coloring problem
- Problem definition : X, D and C
  1. $X = \{X_1, X_2, \ldots, X_n\}$
  2. $D = \{D_1, D_2, \ldots, D_n\}$
  3. C is a set of constraints.
- Consistent assignment
- Complete assignment

Constraint Satisfaction Problem

- Australian map coloring problem
- Problem definition : X, D and C
  1. $X = \{X_1, X_2, \ldots, X_n\}$
  2. $D = \{D_1, D_2, \ldots, D_n\}$
  3. C is a set of constraints.
- Consistent assignment
- Complete assignment
- Solution : consistent and complete

Constraint Satisfaction Problem (CSP)

- constraint graph vs. state space graph

Constraint Satisfaction Problem (CSP)

- constraint graph vs. state space graph
- $X = \{WA, NT, Q, NSW, V, SA, T\}$

Constraint Satisfaction Problem (CSP)

- constraint graph vs. state space graph
- $X = \{WA, NT, Q, NSW, V, SA, T\}$
- $D_i = \{red, green, blue\}$

Constraint Satisfaction Problem (CSP)

- constraint graph vs. state space graph
- $X = \{WA, NT, Q, NSW, V, SA, T\}$
- $D_i = \{red, green, blue\}$
- $C = \{WA \neq NT, WA \neq SA, \ldots, V \neq NSW\}$

Constraint Satisfaction Problem (CSP)

- constraint graph vs. state space graph
- $X = \{WA, NT, Q, NSW, V, SA, T\}$
- $D_i = \{red, green, blue\}$
- $C = \{WA \neq NT, WA \neq SA, \ldots, V \neq NSW\}$
- Multiple solutions are possible.

Constraint Satisfaction Problem (CSP)

- constraint graph vs. state space graph
- $X = \{WA, NT, Q, NSW, V, SA, T\}$
- $D_i = \{red, green, blue\}$
- $C = \{WA \neq NT, WA \neq SA, \ldots, V \neq NSW\}$
- Multiple solutions are possible.
- Goal: understand general approaches for solving CSPs.

Car assembly scheduling

Car assembly scheduling

- $X = \{Axle_F, Axle_B, Wheel_{RF}, Wheel_{RB}, Nuts_{RF}, Nuts_{RB}, Cap_{RF}, Cap_{RB}, Inspect\}$

Car assembly scheduling

- $X = \{Axle_F, Axle_B, Wheel_{RF}, Wheel_{RB}, Nuts_{RF}, Nuts_{RB}, Cap_{RF}, Cap_{RB}, Inspect\}$
- Precedence constraints : $T_1 + \underbrace{d_1} \leq T_2$

Car assembly scheduling

- $X = \{Axle_F, Axle_B, Wheel_{RF}, Wheel_{RB}, Nuts_{RF}, Nuts_{RB}, Cap_{RF}, Cap_{RB}, Inspect\}$

- Precedence constraints : $T_1 + d_1 \leq T_2$
  - $Axle_F + 10 \leq Wheel_{RF}$; $Axle_B + 10 \leq Wheel_{RB}$

Car assembly scheduling

- $X = \{Axle_F, Axle_B, Wheel_{RF}, Wheel_{RB}, Nuts_{RF}, Nuts_{RB}, Cap_{RF}, Cap_{RB}, Inspect\}$

- Precedence constraints : $T_1 + d_1 \leq T_2$
  - $Axle_F + 10 \leq Wheel_{RF}$; $Axle_B + 10 \leq Wheel_{RB}$
  - $Wheel_{RF} + 1 \leq Nuts_{RF}$; $Wheel_{RB} + 1 \leq Nuts_{RB}$

Car assembly scheduling

- $X = \{Axle_F, Axle_B, Wheel_{RF}, Wheel_{RB}, Nuts_{RF}, Nuts_{RB}, Cap_{RF}, Cap_{RB}, Inspect\}$
- Precedence constraints : $T_1 + d_1 \leq T_2$
  - $Axle_F + 10 \leq Wheel_{RF}$; $Axle_B + 10 \leq Wheel_{RB}$
  - $Wheel_{RF} + 1 \leq Nuts_{RF}$; $Wheel_{RB} + 1 \leq Nuts_{RB}$
  - $Nuts_{RF} + 2 \leq Cap_{RF}$

Car assembly scheduling

- $X = \{Axle_F, Axle_B, Wheel_{RF}, Wheel_{RB}, Nuts_{RF}, Nuts_{RB}, Cap_{RF}, Cap_{RB}, Inspect\}$

- Precedence constraints : $T_1 + d_1 \leq T_2$
  - $Axle_F + 10 \leq Wheel_{RF}$; $Axle_B + 10 \leq Wheel_{RB}$
  - $Wheel_{RF} + 1 \leq Nuts_{RF}$; $Wheel_{RB} + 1 \leq Nuts_{RB}$
  - $Nuts_{RF} + 2 \leq Cap_{RF}$
  - $(Axle_F + 10 \leq Axle_B)$ or $(Axle_B + 10 \leq Axle_F)$

Car assembly scheduling

- $X = \{Axle_F, Axle_B, Wheel_{RF}, Wheel_{RB}, Nuts_{RF}, Nuts_{RB}, Cap_{RF}, Cap_{RB}, Inspect\}$
- Precedence constraints : $T_1 + d_1 \leq T_2$
  - $Axle_F + 10 \leq Wheel_{RF}$; $Axle_B + 10 \leq Wheel_{RB}$
  - $Wheel_{RF} + 1 \leq Nuts_{RF}$; $Wheel_{RB} + 1 \leq Nuts_{RB}$
  - $Nuts_{RF} + 2 \leq Cap_{RF}$
  - $(Axle_F + 10 \leq Axle_B)$ or $(Axle_B + 10 \leq Axle_F)$
  - $Cap_{RF} + 1 \leq Inspect$

$Cap_{RB} + 1 \leq Inspect$

Car assembly scheduling

- $X = \{Axle_F, Axle_B, Wheel_{RF}, Wheel_{RB}, Nuts_{RF}, Nuts_{RB}, Cap_{RF}, Cap_{RB}, Inspect\}$
- Precedence constraints : $T_1 + d_1 \leq T_2$
  - $Axle_F + 10 \leq Wheel_{RF}$; $Axle_B + 10 \leq Wheel_{RB}$
  - $Wheel_{RF} + 1 \leq Nuts_{RF}$; $Wheel_{RB} + 1 \leq Nuts_{RB}$
  - $Nuts_{RF} + 2 \leq Cap_{RF}$
  - $(Axle_F + 10 \leq Axle_B)$ or $(Axle_B + 10 \leq Axle_F)$
  - $Cap_{RF} + 1 \leq Inspect$
- $D_i = \{1, 2, 3, \ldots, 27\}$

Types of constraints

- Unary constraint

Types of constraints

- Unary constraint

  $WA \neq Green$

Types of constraints

- Unary constraint

  $WA \neq Green$
- Binary constraint

Types of constraints

- Unary constraint

  $WA \neq Green$

- Binary constraint

  $WA \neq NT$

Types of constraints

- Unary constraint

  $WA \neq Green$
- Binary constraint

  $WA \neq NT$
- Global constraint

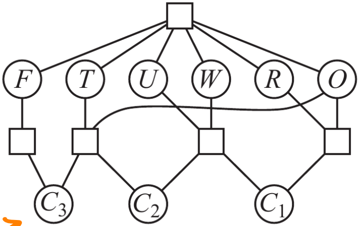
Types of constraints

- Unary constraint
  $WA \neq Green$
- Binary constraint
  $WA \neq NT$
- Global constraint
  $Alldiff(A, B, C, D)$

Cryptoarithmetic Problem

$C_3 \; C_2 \; C_1$

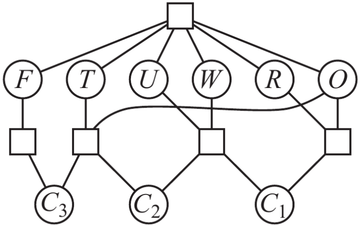
$$\begin{array}{cccc} & T & W & O \\ + & T & W & O \\ \hline F & O & U & R \end{array}$$



- $Alldiff(F, T, U, W, R, O)$

Cryptoarithmetic Problem

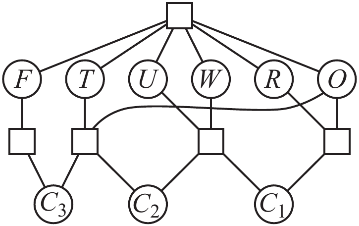
$$\begin{array}{cccc} & T & W & O \\ + & T & W & O \\ \hline F & O & U & R \end{array}$$



- *Alldiff* (F, T, U, W, R, O)
  It is given that all letters are distinct.

Cryptoarithmetic Problem

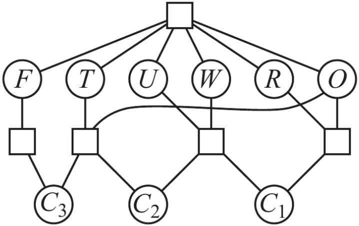
$$\begin{array}{r} T\ W\ O \\ +\ T\ W\ O \\ \hline F\ O\ U\ R \end{array}$$



- *Alldiff*(F, T, U, W, R, O)
  It is given that all letters are distinct.
- $O + O = 10 \times C_1 + R$

Cryptoarithmetic Problem

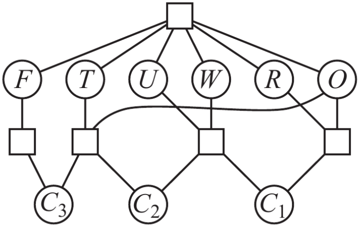
$$
\begin{array}{cccc}
 & T & W & O \\
+ & T & W & O \\
\hline
F & O & U & R
\end{array}
$$



- $Alldiff(F, T, U, W, R, O)$
  It is given that all letters are distinct.
- $O + O = 10 \times C_1 + R$
- $C_1 + W + W = 10 \times C_2 + U$

Cryptoarithmetic Problem

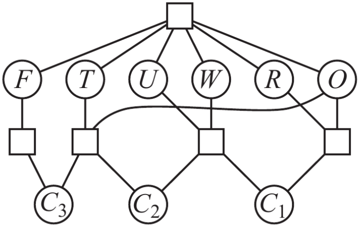
$$\begin{array}{r} T\ W\ O \\ +\ T\ W\ O \\ \hline F\ O\ U\ R \end{array}$$



- $Alldiff(F, T, U, W, R, O)$
  It is given that all letters are distinct.
- $O + O = 10 \times C_1 + R$
- $C_1 + W + W = 10 \times C_2 + U$
- $C_2 + T + T = 10 \times C_3 + O$

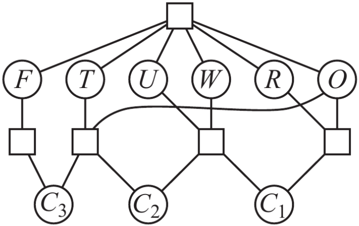
Cryptoarithmetic Problem

$$\begin{array}{cccc} & T & W & O \\ + & T & W & O \\ \hline F & O & U & R \end{array}$$



- $Alldiff(F, T, U, W, R, O)$
  It is given that all letters are distinct.
- $O + O = 10 \times C_1 + R$
- $C_1 + W + W = 10 \times C_2 + U$
- $C_2 + T + T = 10 \times C_3 + O$
- $C_3 = F$

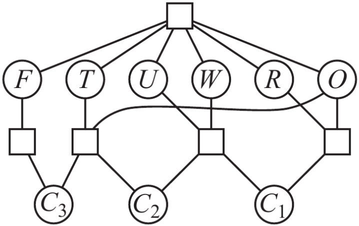
$$\begin{array}{cccc} & T & W & O \\ + & T & W & O \\ \hline F & O & U & R \end{array}$$



- *Alldiff*(F, T, U, W, R, O)
  It is given that all letters are distinct.
- $O + O = 10 \times C_1 + R$
- $C_1 + W + W = 10 \times C_2 + U$
- $C_2 + T + T = 10 \times C_3 + O$
- $C_3 = F$
- $F \neq 0$

Cryptoarithmetic Problem

$$\begin{array}{cccc} & T & W & O \\ + & T & W & O \\ \hline F & O & U & R \end{array}$$



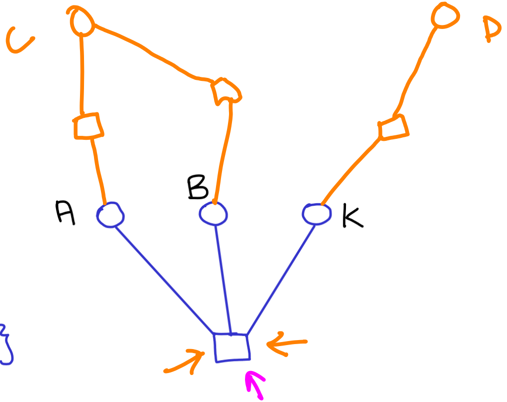
- $Alldiff(F, T, U, W, R, O)$
  It is given that all letters are distinct.
- $O + O = 10 \times C_1 + R$
- $C_1 + W + W = 10 \times C_2 + U$
- $C_2 + T + T = 10 \times C_3 + O$
- $C_3 = F$
- $F \neq 0$
- $D_i = \{0, \ldots, 9\}$

Constraint satisfaction problem (CSP)

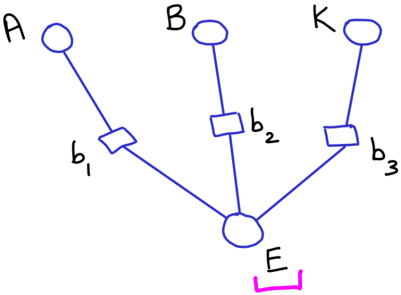
$$X = \{A, B, K\}$$

$$D_i = \{1, 2, 3\}$$

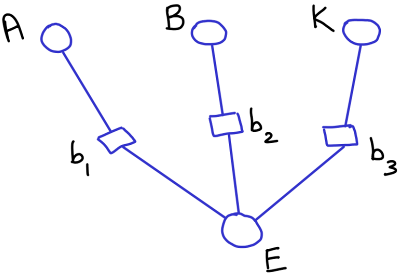
$$C = \{A + B = K\}$$



n-ary constraint to binary constraint



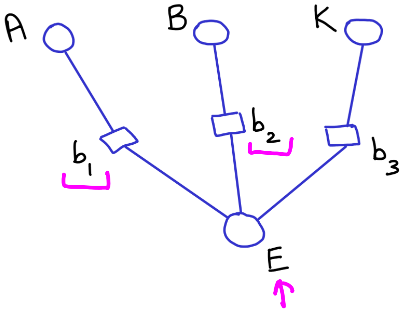
n-ary constraint to binary constraint



3-ary

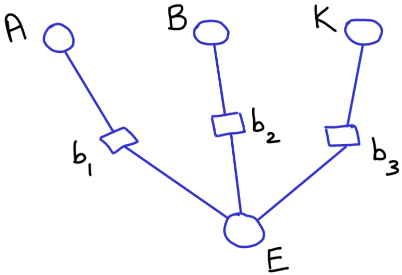
- $\underbrace{D_E}_{} = \{< 1, 2, 3 >, < 2, 1, 3 >, < 1, 1, 2 >\}$

n-ary constraint to binary constraint



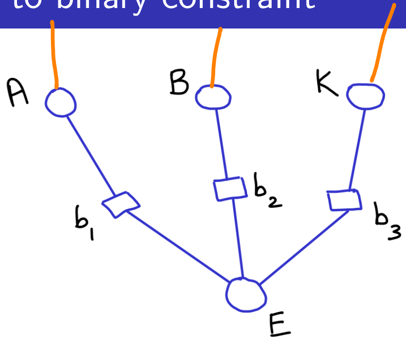
- $D_E = \{< 1,2,3 >, < 2,1,3 >, < 1,1,2 >\}$
- Constraint b_1 checks whether A equals first element of the tuple assigned to E.

n-ary constraint to binary constraint



- $D_E = \{< 1, 2, 3 >, < 2, 1, 3 >, < 1, 1, 2 >\}$
- Constraint b_1 checks whether A equals first element of the tuple assigned to E.
- Constraint b_2 checks whether B equals second element of the tuple assigned to E.

n-ary constraint to binary constraint



- $D_E = \{<1,2,3>, <2,1,3>, <1,1,2>\}$
- Constraint b_1 checks whether A equals first element of the tuple assigned to E.
- Constraint b_2 checks whether B equals second element of the tuple assigned to E.
- Constraint b_3 checks whether K equals third element of the tuple assigned to E.

Dual-graph transformation

- Add a new variable corresponding to each n-ary constraint, where $n > 2$.

Dual-graph transformation

- Add a new variable corresponding to each n-ary constraint, where $n > 2$.
- The domain of the new variable consists of n-tuples whose elements satisfy the corresponding n-ary constraint.

Dual-graph transformation

- Add a new variable corresponding to each n-ary constraint, where $n > 2$.
- The domain of the new variable consists of n-tuples whose elements satisfy the corresponding n-ary constraint.
- Add a binary constraint between the new variable and each of the original n variables (that were participating in the original n-ary constraint).

Finding a solution for a CSP

- The problem : $O(d^n)$ possible assignments.

$d \times d \times \cdots d$

d

Finding a solution for a CSP

- The problem : $O(d^n)$ possible assignments.
- Reduce the number of legal values for different variables.

Finding a solution for a CSP

- The problem : $O(d^n)$ possible assignments.
- Reduce the number of legal values for different variables.
- Achieve $\underbrace{\text{node consistency}}$

  $\underbrace{WA \neq Green}$

$D_{WA} = \{r, \cancel{g}, b\}$

Finding a solution for a CSP

- The problem : $O(d^n)$ possible assignments.
- Reduce the number of legal values for different variables.
- Achieve node consistency

  $WA \neq Green$
- Achieve arc consistency