Factor Graph and Constraint Satisfaction Problems (CSPs) 1

Hwanjo Yu

POSTECH

http://di.postech.ac.kr/hwanjoyu

Search problem

Markov decision processes

Reinforcement learning & games

Constraint satisfaction problems

Markov networks

Bayesian networks

Reflex **States** **Variables** **Logic**

"Low-level intelligence" "High-level intelligence"

Machine learning

State-based models

[Modeling]

Framework	search problems	MDPs/games
Objective	minimum cost paths	maximum value policies

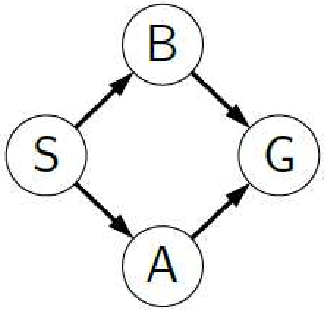
[Algorithms]

Tree-based	backtracking	minimax/expectimax
Graph-based	DP, UCS, A*	value/policy iteration

[Learning]

Methods	structured Perceptron	Q-learning, TD learning

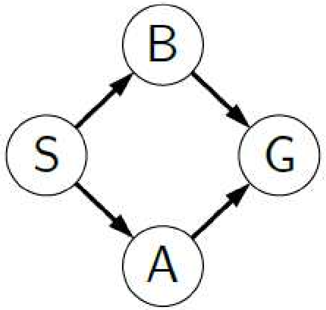
State-based models: takeaway 1



Key idea: specify locally, optimize globally

- Modeling: specifies local interactions (e.g., action cost or reward)
- Algorithms: find globally optimal solutions (e.g., finding minimum cost paths)

State-based models: takeaway 2



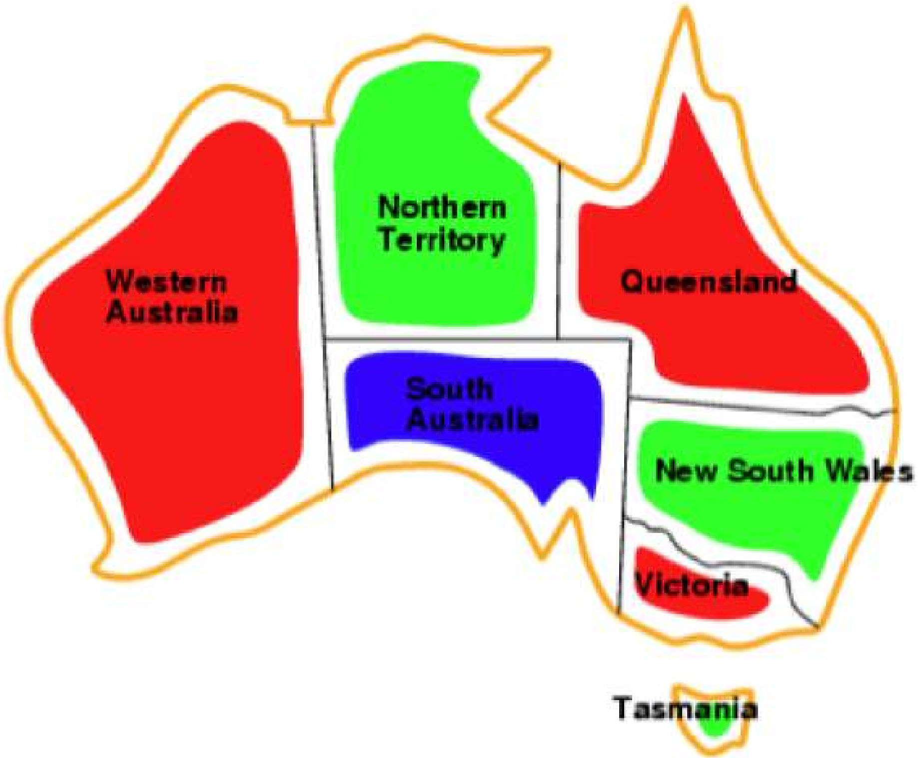
Key idea: state

- A **state** is a summary of all the past actions sufficient to choose future actions **optimally**.
- Thinking about taking a sequence of actions where order is important.
- In some tasks, order is irrelevant => variable-based models

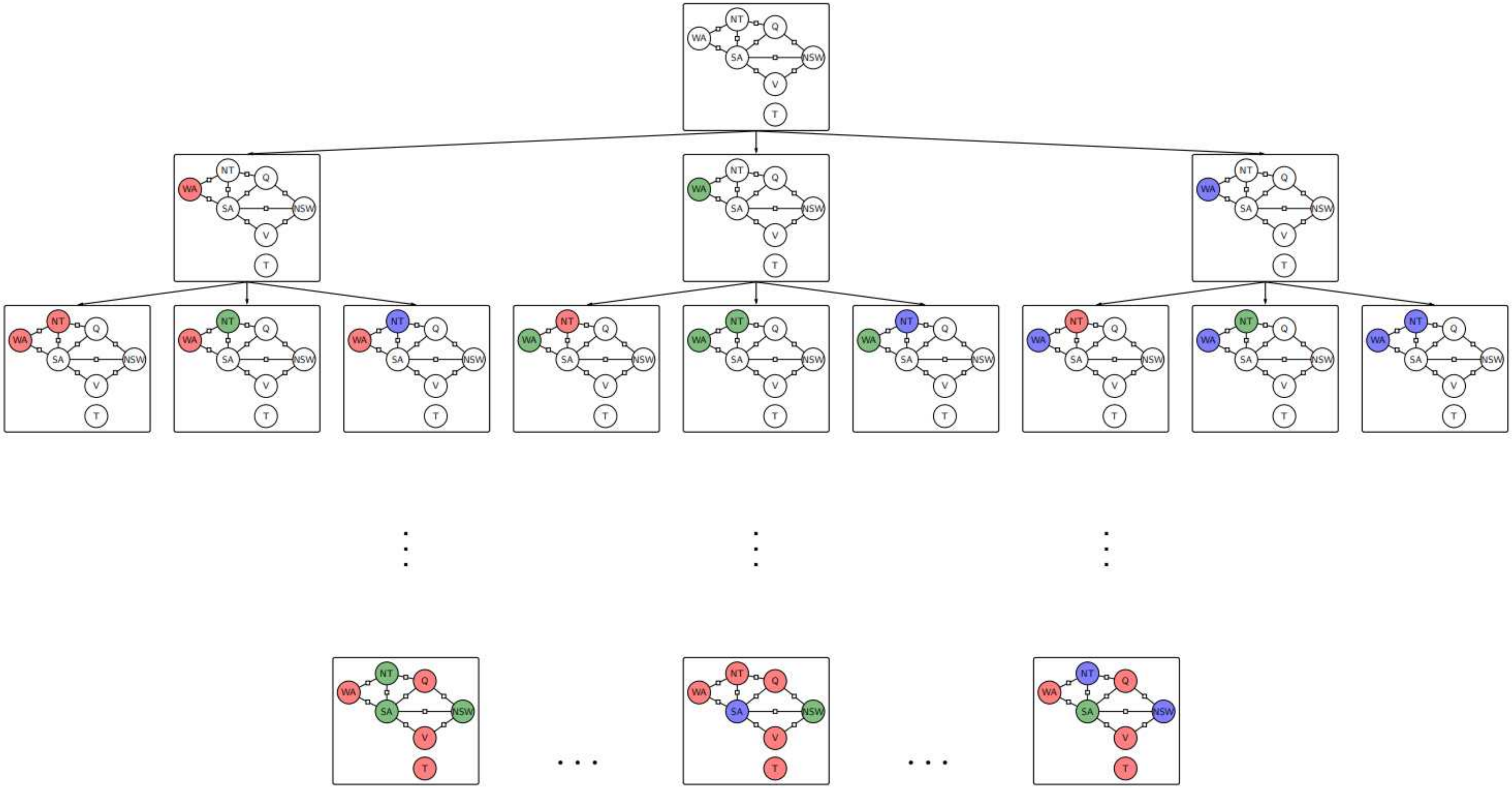
Map coloring

Question: how can we color each of the 7 provinces {red, green, blue} so that no two neighboring provinces have the same color?

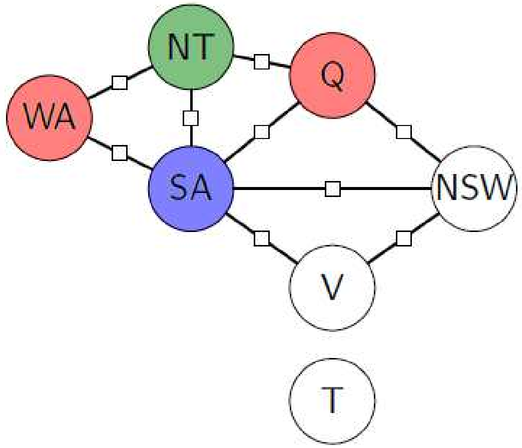
Map coloring



(one possible solution)


WA
NT
Q
SA
NSW
V
T

As a search problem



- State: partial assignment of colors to provinces
- Action: assign next uncolored province a compatible color

Can we do better? Exploit the problem structure!

- Variable ordering doesn't affect correctness => choose a better ordering by "lookahead" rather than trying all orderings
- Variables are interdependent in a local way => Tasmania is independent. Do something!

Variable-based models

Variable-based models (or graphical models):

- Constraint satisfaction problems (CSP)
- Probabilistic Graphical Model (PGM): Markov networks (undirected graphical model), Bayesian networks (directed graphical model), HMMs, CRFs, etc.

- Problem => assignments of values to variables (modeling)
- How to find the assignment? => algorithm

Applications

- Delivery/routing: how to assign packages to trucks to deliver to customers
- Sports scheduling: when to schedule pairs of teams to minimize travel
- Formal verification: ensure circuit/program works on all inputs

Roadmap

Modeling

Definitions

Examples

Backtracking (exact) search

Dynamic ordering

Arc consistency

Approximate search

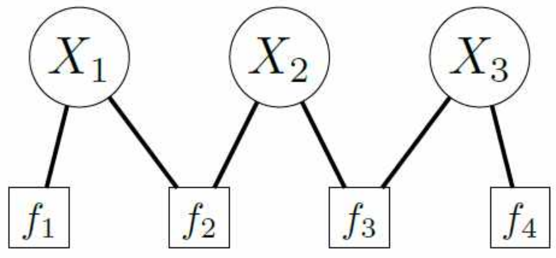
Beam search

Local search

Factor graph example: voting

Three people X_1, X_2, X_3 vote for a color, and we know that…

- X_1 must have B (f_1)
- X_1 and X_2 must have the same color (f_2)
- X_2 and X_3 weakly prefer to have the same color (f_3)
- X_3 is leaning toward R (f_4)



x_1	$f_1(x_1)$
R	0
B	1

$f_1(x_1) = [x_1 = \text{B}]$

x_1	x_2	$f_2(x_1, x_2)$
R	R	1
R	B	0
B	R	0
B	B	1

x_2	x_3	$f_3(x_2, x_3)$
R	R	3
R	B	2
B	R	2
B	B	3

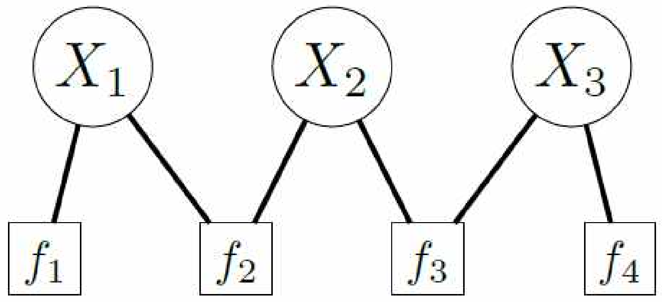
x_3	$f_4(x_3)$
R	2
B	1

$f_4(x_3) = [x_3 = \text{R}] + 1$

$f_2(x_1, x_2) = [x_1 = x_2]$ $\quad f_3(x_2, x_3) = [x_2 = x_3] + 2$

- A **variable X_i** assigns a value from **Domain** (e.g., R or B).
- A **factor f_i** (a table or a function) assigns a non-negative number describing how good the assignment to a subset of the variables is.

Factor graph



Definition: factor graph

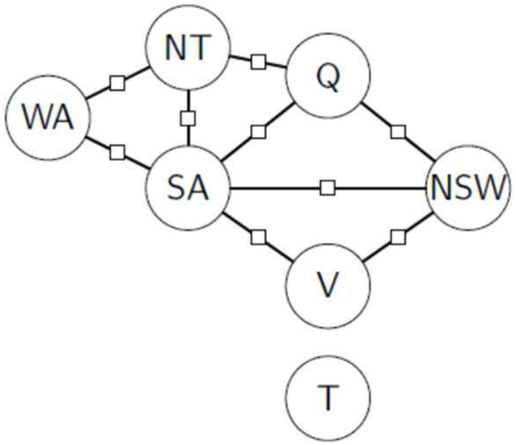
- Variables:

$$X = (X_1, \dots, X_n), \text{ where } X_i \in \text{Domain}_i$$

- Factors:

$$f_1, \dots, f_m, \text{ with each } f_j(X) \geq 0$$

Example: map coloring



Variables:

- $X = (\text{WA}, \text{NT}, \text{SA}, \text{Q}, \text{NSW}, \text{V}, \text{T})$
- $\text{Domain}_i \in \{\text{R}, \text{G}, \text{B}\}$

Factors:

- $f_1(X) = [\text{WA} \neq \text{NT}]$ ([...] is the indicator function.)
- $f_2(X) = [\text{NT} \neq \text{Q}]$
- ...

Factors

Definition: scope and arity

- **Scope** of a factor f_j : set of variables it depends on.
- **Arity** of f_j : the number of variables in the scope.
- **Unary** factors (arity 1); **Binary** factors (arity 2).
- **Constraints** are factors that return 0 or 1.

Example: map coloring

- Scope of $f_1(X) = [\mathrm{WA} \neq \mathrm{NT}]$ is $\{\mathrm{WA}, \mathrm{NT}\}$
- f_1 is a binary factor

Assignment weights example: voting

x_1	$f_1(x_1)$
R	0
B	1

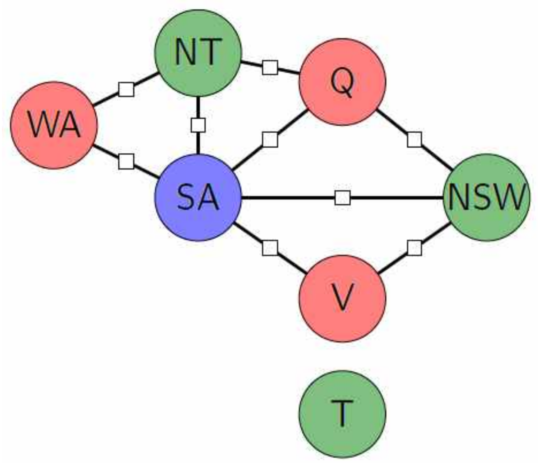
x_1	x_2	$f_2(x_1, x_2)$
R	R	1
R	B	0
B	R	0
B	B	1

x_2	x_3	$f_3(x_2, x_3)$
R	R	3
R	B	2
B	R	2
B	B	3

x_3	$f_4(x_3)$
R	2
B	1

x_1	x_2	x_3	Weight
R	R	R	$0 \cdot 1 \cdot 3 \cdot 2 = 0$
R	R	B	$0 \cdot 1 \cdot 2 \cdot 1 = 0$
R	B	R	$0 \cdot 0 \cdot 2 \cdot 2 = 0$
R	B	B	$0 \cdot 0 \cdot 3 \cdot 1 = 0$
B	R	R	$1 \cdot 0 \cdot 3 \cdot 2 = 0$
B	R	B	$1 \cdot 0 \cdot 2 \cdot 1 = 0$
B	B	R	$1 \cdot 1 \cdot 2 \cdot 2 = 4$
B	B	B	$1 \cdot 1 \cdot 3 \cdot 1 = 3$

Example: map coloring



Assignment: $x = \{\text{WA: R, NT: G, SA: B, Q: R, NSW: G, V: R, T: G}\}$

Weight: $\text{Weight}(x) = 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 = 1$

Assignment: $x' = \{\text{WA: R, NT: R, SA: B, Q: R, NSW: G, V: R, T: G}\}$

Weight: $\text{Weight}(x') = 0 \cdot 0 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 = 0$

*all the factors are multiplied (not added) thus any factor has veto power.

Assignment weights (product of all factors)

Definition: assignment weight

- Each **assignment** $x = (x_1, \dots, x_n)$ has a **weight**:

$$\text{Weight}(x) = \prod_{j=1}^{m} f_j(x)$$

Objective: find the maximum weight assignment

$$arg \ \max_x \text{Weight}(x)$$

Constraint satisfaction problems

Definition: constraint satisfaction problem (CSP)

- A CSP is a factor graph where all factors are **constraints**:

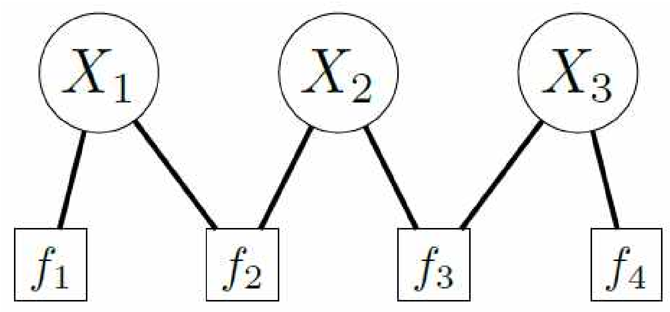
$$f_j(x) \in \{0,1\} \text{ for all } j = 1, \dots, m$$

- The constraint is satisfied iff $f_j(x) = 1$.

Definition: consistent assignments

- An assignment x is **consistent** iff $\mathrm{Weight}(x) = 1$ (i.e., **all** constraints are satisfied).

Summary so far



Factor graph (general)	**CSP** (all or nothing)
variables	variables
factors	constraints
maximum weight assignment	consistent assignment

Example: LSAT question

Three sculptures (A, B, C) are to be exhibited in rooms 1, 2 of an art gallery.

The exhibition must satisfy the following conditions:

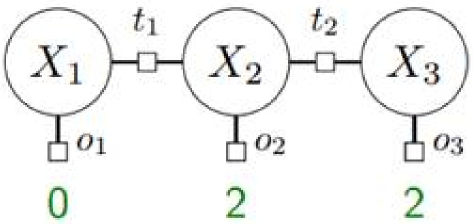
- Sculptures A and B cannot be in the same room.
- Sculptures B and C must be in the same room.
- Room 2 can only hold one sculpture.

Example: object tracking

- (O) Noisy sensors report positions: 0, 2, 2.
- (T) Objects can't teleport.

What trajectory did the object take?

Factor graph:



x_1	$o_1(x_1)$
0	2
1	1
2	0

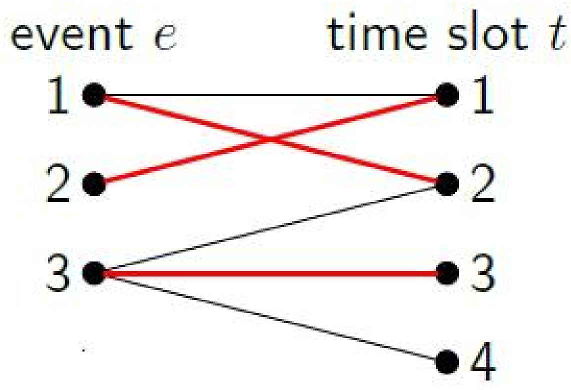
x_2	$o_2(x_2)$
0	0
1	1
2	2

x_3	$o_3(x_3)$
0	0
1	1
2	2

$\|x_i - x_{i+1}\|$	$t_i(x_i, x_{i+1})$
0	2
1	1
2	0

- Variables $X_i \in \{0,1,2\}$: position of object at time i
- Observation factors $o_i(x_i)$: noisy information compatible with position
- Transition factors $t_i(x_i, x_{i+1})$: object positions can't change too much

Example: event scheduling



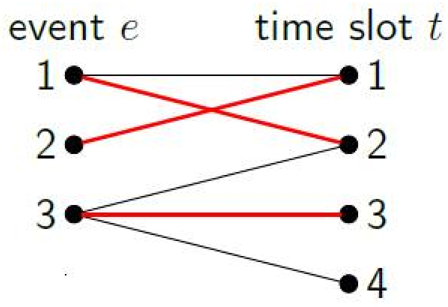
Have E events and T time slots

- (C1) Each event e must be put in **exactly one** time slot
- (C2) Each time slot t can have **at most one** event
- (C3) Event e allowed in time slot t only if $(e, t) \in A$

Example: event scheduling (formulation 1)

Have E events and T time slots

- (C1) Each event e must be put in **exactly one** time slot
- (C2) Each time slot t can have **at most one** event
- (C3) Event e allowed in time slot t only if $(e, t) \in A$



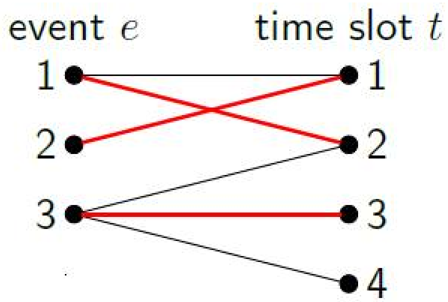
CSP formulation 1:

- Variables: for each event $e, X_e \in \{1, \ldots, T\}$; satisfies (C1)
- Constraints (only one event per time slot): for each pair of events $e \neq e'$, enforce $[X_e \neq X_{e'}]$; satisfies (C2)
  => How many factors?
  - E.g., $\{X_1: 1, X_2: 1, X_3: 3\}$ is bad because $\{X_1 = X_2\}$
- Constraints (only schedule allowed times): for each event e, enforce $[(e, X_e) \in A]$; satisfies (C3)
  - E.g., $\{X_1: 1, X_2: 4, X_3: 3\}$ is bad because $\{(2,4) \text{ not in } A\}$

Example: event scheduling (formulation 2)

Have E events and T time slots

- (C1) Each event e must be put in **exactly one** time slot
- (C2) Each time slot t can have **at most one** event
- (C3) Event e allowed in time slot t only if $(e, t) \in A$



CSP formulation 2:

- Variables: for each time slot $t, Y_t \in \{1, \ldots, E\} \cup \{\emptyset\}$; satisfies (C2)
- Constraints (each event is scheduled exactly once): for each event e, enforce $[Y_t = e \text{ for exactly one } t]$; satisfies (C1)
- Constraints (only schedule allowed times): for each time slot t, enforce $[Y_t = \emptyset \text{ or } (Y_t, t) \in A]$; satisfies (C3)

Example: program verification

```
def foo(x, y):
    a = x * x
    b = a + y * y
    c = b - 2 * x * y
    return c
```

Specification: $c \geq 0$ for all x and y

CSP formulation:

- Variables: x, y, a, b, c
- Constraints (program statements): $[a = x^2], [b = a + y^2], [c = b - 2xy]$ (Note: program is assignment, and CSP is mathematical equality)
- Constraint (negation of specification): $[c < 0]$

Program satisfies specification iff CSP has no consistent assignment.

Summary: modeling CSP

- Decide on variables and domains
- Translate each desideratum into a set of factors
- Try to keep CSP small (variables, factors, domains, arities)
- When implementing each factor, think in terms of checking a solution rather than computing the solution