

# Asignación de Salas



Queremos asignar las salas y horarios para los cursos de la PUC:

- ☐ Los horarios deben coincidir con los módulos
- ☐ En una misma sala no se pueden dictar dos cursos a la vez
- ☐ Un profesor no puede dictar dos cursos a la vez

¿Cómo abordamos un problema como este?



Problemas como este se llaman de **satisfacción de restricciones**

Es una familia entera de problemas con las mismas características

¿Qué tan rápido podrán resolverse los **CSP**?

# SAT

Sea  $\varphi$  una fórmula en lógica proposicional

$\varphi$  se dice **satisfacible** si existe forma de hacerla verdadera

Averiguar si  $\varphi$  es **satisfacible** es **NP-Completo**

# SAT como CSP



Queremos encontrar una asignación a cada variable de  $\varphi$ :

- ❑ La fórmula  $\varphi$  debe hacerse verdadera

¿Qué nos dice esto sobre los **CSP**?

# Resolver CSPs



Una forma es generar todas las permutaciones posibles

¿Es posible hacerlo mejor?

Quizás no es necesario generar **todas** las permutaciones...

# CSPs en general

Dadas variables  $x_1, \dots, x_n$  con dominios  $d_1, \dots, d_n$

Y un set de restricciones  $R$

Encontrar una asignación para cada  $x$  que respete  $R$

# ¿Es posible?



Dado un problema, ¿es posible resolverlo?

La idea es responder esa pregunta recursivamente

Si asignamos una variable, ¿qué nos queda?

*asignar salas y horarios*( $C, S, i$ ):

*if*  $i = |C|$ , *return true*

$c = C_i$

*for*  $m \in \text{Módulos}$ :

*if*  $c.\text{profesor}$  está ocupado al módulo  $m$ , *continue*

*for*  $s \in S$ :

*if*  $s$  está ocupada en el módulo  $m$ , *continue*

Asignar clase  $c$  al horario  $m$  y sala  $s$

*if* *asignar salas y horarios*( $C, S, i + 1$ ):

*return true*

Desasignar clase  $c$  al horario  $m$  y sala  $s$

*return false*



# Backtracking

Esta estrategia se conoce como **backtracking**

La idea es **descartar** permutaciones que violan alguna restricción

Eso significa que **siempre** es igual o más rápido que fuerza bruta

*is solvable*( $X, D, R$ ):

*if*  $X = \emptyset$ , *return true*

$x \leftarrow$  alguna variable de  $X$

*for*  $v \in D_x$ :

*if*  $x = v$  viola  $R$ , *continue*

$x \leftarrow v$

*if is solvable*( $X - \{x\}, D, R$ ):

*return true*

$x \leftarrow \emptyset$

*return false*

# Modelación

Para resolver un problema siempre es necesario:

- Identificar las componentes del problema
- Expresarlas en términos de variables y restricciones
- Preocuparse de que las **operaciones** sean eficientes

¡De no hacerlo bien, no estaríamos ganando nada!

# N-Queens



En un tablero de ajedrez de  $n \times n$  se quieren poner  $n$  reinas:

- Ninguna reina debe poder atacar a otra reina

¿Cómo modelamos esto?

# Sudoku



¿Cuáles son las variables en el sudoku?

¿Cuáles son sus dominios?

¿Cuáles son las restricciones?