1. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:ColumbiaX+CSMM.101x+3T2020+type@vertical+block@795ac52b50894c94ba5ffce486a10f00?show_title=0&show_bookmark_button=0#transcript-end-fffb099a7e1b4b4698f5e67391ff00f3)
2. Welcome back, and congratulations on making it up
3. to this point of the course.
4. I really hope that you're having an enjoyable learning
5. experience.
6. It has been a learning experience for me as well.
7. I'm not used to not seeing my students during the lectures,
8. but hopefully I'm getting the hang of it.
9. So today, we will talk about Constraint Satisfaction
10. Problems, also known CSPs.
11. CSPs are actually search problems
12. but specific search problems.
13. Recall from the previous lectures
14. that search algorithms care about finding
15. the sequence of actions that lead to the goal.
16. So we care about the sequence of actions about the goal,
17. but we also care about finding the least-cost solution,
18. for example, or finding the solutions in few steps.
19. We often use some search algorithms like BFS, DFS, et
20. cetera, and we use even some heuristics, or rule of thumbs,
21. to speed up the search and to find the optimal solution.
22. CSPs are search problems, too, in which we don't care much
23. about the path or the sequence of actions
24. that lead to the goal as we care about the goal itself.
25. So, specifically, remember from the previous lectures
26. we spoke about different representations of states.
27. So we talked about atomic representations
28. that is used in search problems in which we want
29. to go from point B to point C. And each state is,
30. in this case, a black box.
31. We don't have any representation details or presentation
32. of these states.
33. All we want to do is to reach the goal
34. and find the sequence of actions that lead to the goal.
35. In constraint satisfaction problems
36. the state is not just a black box.
37. The state has a more complex representation to it
38. that allows for smarter and deeper understanding
39. of the problem at hand in order to solve it.
40. More specifically, the state in a CSP
41. is defined by variables Xi with values from some domain Di.
42. Recall from the previous lecture that we called this a factored
43. representation in which we have some variables defining
44. the features of the state.
45. A goal test, in this case, is a set
46. of constraints specifying allowable combinations
47. of variable values.
48. Just to put things back in context,
49. CSP falls under the representations of variables,
50. so remember this axis about low intelligence
51. to high intelligence levels.
52. So we have seen search problems and adversarial games.
53. We also have seen several machining learning methodologies.
54. So, basically, CSPs fall under the variable representation model
55. in which we are going toward the high level of intelligence
56. systems.
57. So let's now define what is a CSP problem.
58. So a CSP problem consists of three elements.
59. And these are a set of variables, X,
60. that's equal to the set X1, X2, Xn, if you have n variables--
61. a set of domains for each variable, D1, D2, Dn--
62. and a set of constraints, C, that
63. specify allowable combinations of value.
64. So what kind of values can we combine together
65. to solve that CSP?
66. Solving the CSP causes, then, in finding the assignments that
67. actually satisfy all the constraints of the problem.
68. We learn different concepts in this lecture.
69. And these include how to formalize the problem,
70. how to do a backtracking search in CSPs,
71. how to check for arc consistency,
72. among other concepts.
73. Finally, we call the solution a consistent assignment.
74. Why consistent?
75. Because we try to find the assignment
76. that does not violate the constraint of the problem.
77. Let's take an example of CSP called Map coloring.
78. Map coloring is inspired by graph theory.
79. It's a famous problem in graph theory
80. in which you want to color a map and not use the same color
81. on any two adjacent regions.
82. So let's move now from North America
83. to Australia, in which we are given a map,
84. and this map has different regions of Australia,
85. seven regions specifically, right.
86. And you want to be able to color this message so
87. that we don't use the same color on two adjacent regions.
88. To formalize the problem as a CSP,
89. we are going to first define what are the variables.
90. And the variables in this case are
91. x equal to Western Australia, WA, Northern Territory, NT, et
92. cetera.
93. So we have seven variables in our problem.
94. The domain are the colors you are given to color the map.
95. And these are red, green, and blue,
96. which means that any variable in x
97. can take any of these three values.
98. And finally, the constraints are: adjacent regions
99. must have different colors.
100. So could spell this constraint as follows.
101. We could say, for example, we want WA to be different of NT,
102. or we could write it as a pair.
103. We want the pair WA, NT to belong to the set
104. either with green, or red blue, or blue green, et cetera.
105. So we're going to spell out all possible pairs
106. such as the first element of the pair
107. is different from the second element of the pair.
108. This is a typical CSP problem.
109. And the solution for this kind of CSP
110. is to find this assignment of colors to the regions.
111. Here's an example.
112. So an example of solutions to this
113. would be a possible solution is equal to what?
114. Give the color red to WA, the color
115. green to Northern Territory, red to Queensland, et cetera.
116. So note here that we could color Tasmania
117. in another color, any other color,
118. because it's not linked to any other area on the map.
119. So finding a solution for CSP is to find an assignment of colors
120. to the different variables in the problem.
121. Real-world problems of CSPs are numerous,
122. and this includes assignment problems.
123. For example, who teaches what class?
124. Timetabling problems such as which
125. class is offered when and where, hardware configurations,
126. spreadsheets, transportation scheduling,
127. factory scheduling, floor planning.
128. And notice that many real-world problems
129. involve real value variables rather than discrete variables
130. such as the color problem in the map coloring.
131. It's often useful to represent the CSP
132. as what we call the constraint graph, in which we have nodes
133. representing the variables and edges representing
134. the constraints between the nodes.
135. So, for example, in the Australia map,
136. we could represent each of these regions
137. as a node, and any constraint we have has actually an edge
138. between those nodes.
139. So, for example, we have the constraint
140. that WA should be different of NT.
141. A is represented by this node here.
142. We talk about binary CSPs when constraints relate at
143. most one pair of variables or two variables.
144. It's often CSP algorithms which leverage
145. that structure or that graph that represents the problem
146. to speed up the search.
147. For example, Tazmania is independent
148. of the other regions, and we can color it with whatever color
149. we want among the three colors, red, blue, and green.
150. So using this graph is very interesting
151. to speed up the search and come up
152. with better search algorithms to solve the problem.
153. We also have a variety of variables,
154. and this includes discrete variables
155. or continuous variables.
156. So in discrete variables we could have finite domain
157. or infinite domain.
158. And in finite domain, we assume that we have n variables
159. and each variable takes its value in d values,
160. then the number of complete assignments we can have
161. is on the order of d to the n.
162. Examples of this kind of variables
163. include map coloring, the 8-queen problem,
164. and so on and so forth.
165. If the domain is infinite, such as integers or strings,
166. but it's still discrete in the values,
167. then we need to use some constraint language.
168. The constraint language we define actually
169. how we spell out the constraint of the problem.
170. For job scheduling, for example, to express that time 2 needs
171. to start D after time 1, then we could
172. express that T1 plus D is less than or equal to T2.
173. If the variables are continuous, which
174. is common in operating research, then we
175. use some techniques like linear programming,
176. with linear and nonlinear equalities
177. to solve the problem.
178. We may also have a variety of constraints.
179. And this includes unary constraints which
180. involve only one variable.
181. For example, we could say that SA is different than green,
182. is the one variable different than a value?
183. We could have binary constraint, which is most common,
184. in which we have a constraint on pairs of variables such as SA
185. different than WA means that the color of SA
186. different than the color of WA.
187. We may also have what we call global constraints, that
188. involves three or more variables.
189. For example, and a famous one is Alldiff (All Different),
190. that specifies that all the variables
191. must have a different value.
192. Example of this kind of constraint
193. includes cryptoarithmetic puzzles,
194. which we will see in a second, and Sudoku problems.
195. We may also have more subtle kinds of constraints,
196. and this includes preferences.
197. For example, red is better than blue or professor A prefers
198. to teach in the morning.
199. So these are softer, we are going
200. to often represent that by cost for each variable assignment.
201. And this actually includes problems
202. such as constraint optimization problems,
203. in which we are going to include that kind of constraints.
204. Now let's revise again the 8-queen problem
205. seen a few weeks ago.
206. So remember we want to place 8 queens on an eight-
207. by-eight chess board so as no queen attacks another one.
208. And if a queen attacks another one,
209. if it's put on the same column as the other queen,
210. or on the same line, or on the same diagonal.
211. So we have seen this problem as a search problem in which we
212. were looking for the sequence of actions
213. putting a queen after a queen on the board
214. as we solve the problem.
215. So let's discuss this problem now in the CSP framework.
216. A first problem formalization could
217. be that we define one variable per queen.
218. So we have Q1, Q2, Q8, these are the variables.
219. These are the variables X. So X would
220. be equal to queen 1, queen 2, queen 8
221. where each variable would have a value between 1 and 64,
222. this is the size of the board.
223. So the first one would be put at cell one
224. and the last one would be at cell 64.
225. So you're going to start from the top-left corner
226. to the bottom-right corner.
227. A solution to that would be the position of the queen
228. or the assignment of the value between 1 and 64
229. to each of the variables, Q1 to Q8.
230. So a solution could be, queen 1 is 1, queen 2 is 13,
231. queen 3 is 24, and et cetera.
232. A second possibility to formalize the problem
233. is to use still eight variables, Q1, Q2, Q8, but in this case
234. a domain would be more restricted.
235. Each variable could have a value between 1 and 8,
236. considering the columns.
237. In other words, we're going to see
238. in what position in the column we're going to put the queen.
239. So the first one would be in position one.
240. In the second column we will use position 7.
241. And the third column is going to be positioned 5, et cetera.
242. So the domain now for each variable,
243. given that we're going column by column from left to right,
244. it's clear that we're going to just provide
245. the position of the line in that column.
246. So in this case we're going to use less possible values
247. to assign to the variables.
248. We might be tempted to do a brute force search.
249. In other words, can we simply generate and test
250. all possible configurations.
251. You know, just say whether-- check the constraint
252. on each possible configuration and pick the one that
253. satisfies all the constraints.
254. So this might look easy, especially
255. on a four-by-four chess board.
256. So suppose we have a four-queen problem.
257. So in the four-queen problem, if we
258. choose the second formalization of the problem in which case
259. we put one queen per column, so each variable,
260. each queen would have a value between 1 and 8,
261. then it may sound easy.
262. So this is a random generation of all possible queens
263. on the chess board, a four-by-four,
264. but actually we have to be careful
265. that these kinds of configurations are actually
266. deemed to be not successful.
267. So let's suppose that we are going
268. to put a queen on each column.
269. In this case, the first column could have four possibilities
270. to put the first queen, the second
271. would have another four possibilities
272. to put the second queen, the third one we have four
273. possibilities, and the fourth one we
274. have four possibilities, which will give us 4 to the 4,
275. which would be equal to 256.
276. So for a 4-queen problem it's easy to just generate
277. these 256 configurations and test all of them on a computer.
278. It's going to be very fast.
279. However, the question is, is this brute force search
280. scalable?
281. For example, if we move to an 8-queen problem,
282. we don't have any more a choice between four positions
283. on the column, we have eight.
284. So it's going to be 8 times 8 times 8, which
285. is 8 to the 8, which actually will grow really, really big
286. and up to something like 16.7 million
287. possible configurations.
288. The problem becomes very big, and it's
289. hard to really think about generating all possibilities
290. or all configurations and simply test the constraints to pick
291. a solution.
292. So think also about maybe 16-queen
293. in which we will have something like 10
294. to the 20 possible configurations of the boards.
295. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:ColumbiaX+CSMM.101x+3T2020+type@vertical+block@795ac52b50894c94ba5ffce486a10f00?show_title=0&show_bookmark_button=0#transcript-start-fffb099a7e1b4b4698f5e67391ff00f3)