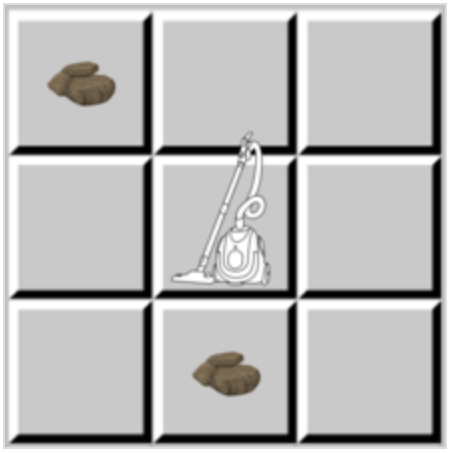
# State space (computer science)

(Redirected from State space)

In computer science, a **state space** is a discrete space representing the set of all possible configurations of a "system".[1] It is a useful abstraction for reasoning about the behavior of a given system and is widely used in the fields of artificial intelligence and game theory.

For instance, the toy problem Vacuum World has a discrete finite state space in which there are a limited set of configurations that the vacuum and dirt can be in. A "counter" system, where states are the natural numbers starting at 1 and are incremented over time[2] has an infinite discrete state space. The angular position of an undamped pendulum[3] is a continuous (and therefore infinite) state space.



Vacuum World, a shortest path problem with a finite state space

## Definition

State spaces are useful in computer science as a simple model of machines. Formally, a state space can be defined as a tuple [*N*, *A*, *S*, *G*] where:

- *N* is a set of states
- *A* is a set of arcs connecting the states
- *S* is a nonempty subset of *N* that contains start states
- *G* is a nonempty subset of *N* that contains the goal states.

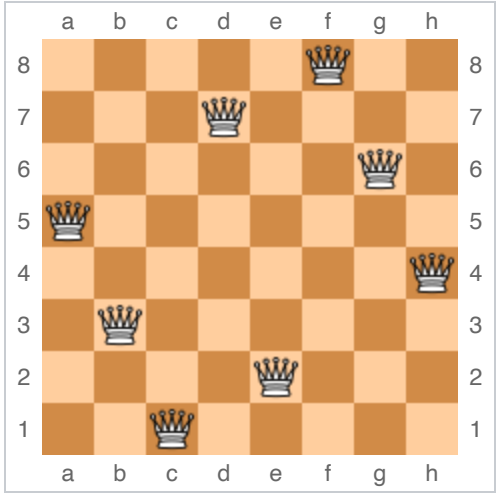
## Properties

A state space has some common properties:

- complexity, where branching factor is important
- structure of the space, see also graph theory:
  - directionality of arcs
  - tree
  - rooted graph

For example, the Vacuum World has a branching factor of 4, as the vacuum cleaner can end up in 1 of 4 adjacent squares after moving (assuming it cannot stay in the same square nor move diagonally). The arcs of Vacuum World are bidirectional, since any square can be reached from any adjacent square, and the state space is not a tree since it is possible to enter a loop by moving between any 4 adjacent squares.

State spaces can be either infinite or finite, and discrete or continuous.



A valid state in the state space of the eight queens puzzle

### Size

The size of the state space for a given system is the number of possible configurations of the space.[3]

#### Finite

If the size of the state space is finite, calculating the size of the state space is a combinatorial problem.[4] For example, in the Eight queens puzzle, the state space can be calculated by counting all possible ways to place 8 pieces on an 8x8 chessboard. This is the same as choosing 8 positions without replacement from a set of 64, or

$$\binom{64}{8} = 4,426,165,368$$

This is significantly greater than the number of legal configurations of the queens, 92. In many games the effective state space is small compared to all reachable/legal states. This property is also observed in Chess, where the effective state space is the set of positions that can be reached by game-legal moves. This is far smaller than the set of positions that can be achieved by placing combinations of the available chess pieces directly on the board.

#### Infinite

All continuous state spaces can be described by a corresponding continuous function and are therefore infinite.[3] Discrete state spaces can also have (countably) infinite size, such as the state space of the time-dependent "counter" system,[2] similar to the system in queueing theory defining the number of customers in a line, which would have state space {0, 1, 2, 3, ...}.

# Exploration

Exploring a state space is the process of enumerating possible states in search of a goal state. The state space of Pacman, for example, contains a goal state whenever all food pellets have been eaten, and is explored by moving Pacman around the board.[5]

### Search States

A search state is a compressed representation of a world state in a state space, and is used for exploration. Search states are used because a state space often encodes more information than is necessary to explore the space. Compressing each world state to only information needed for exploration improves efficiency by reducing the number of states in the search.[5] For example, a state in the Pacman space includes information about the direction Pacman is facing (up, down, left, or

right). Since it does not cost anything to change directions in Pacman, search states for Pacman would not include this information and reduce the size of the search space by a factor of 4, one for each direction Pacman could be facing.

### Methods

Standard search algorithms are effective in exploring discrete state spaces. The following algorithms exhibit both completeness and optimality in searching a state space.[5][6]

- Breadth-First Search
- A* Search
- Uniform Cost Search

These methods do not extend naturally to exploring continuous state spaces. Exploring a continuous state space in search of a given goal state is equivalent to optimizing an arbitrary continuous function which is not always possible, see mathematical optimization.

## See also


- Phase space for information about phase state (like continuous state space) in physics and mathematics.
- Probability space for information about state space in probability.
- Game complexity theory, which relies on the state space of game outcomes
- Cognitive Model#Dynamical systems for information about state space with a dynamical systems model of cognition.
- State space planning
- State (computer science)
- Artificial intelligence
- Dynamical systems
- Glossary of artificial intelligence
- Machine learning
- Mathematical optimization
- Multi-agent system
- Game theory
- Combinatorics

## References

1. Nykamp, Duane. "State space definition" (https://mathinsight.org/definition/state_space). *Math Insights*. Retrieved 17 November 2019.
2. Papernick, Norman. "Infinite States and Infinite State Transitions" (https://www.cs.cmu.edu/afs/cs/academic/class/15671-f95/www/handouts/sm-basics/node8.html). Carnegie Mellon University. Retrieved 12 November 2019.
3. Nykamp, Duane. "The idea of a dynamical system" (https://mathinsight.org/dynamical_system_idea). *Math Insights*. Retrieved 12 November 2019.

4. Zhang, Weixong (1999). *State-space search: algorithms, complexity, extensions, and applications* (https://books.google.com/books?id=bifeGGCDlWcC). Springer. ISBN 978-0-387-98832-0.
5. Abbeel, Pieter. "Lecture 2: Uninformed Search" (http://ai.berkeley.edu/slides/Lecture%202%20--%20Uninformed%20Search/SP14%20CS188%20Lecture%202%20--%20Uninformed%20Search.pptx). *UC Berkeley CS188 Intro to AI*. Retrieved 30 October 2019.
6. Abbeel, Pieter. "Lecture 3: Informed Search" (http://ai.berkeley.edu/slides/Lecture%202%20--%20Uninformed%20Search/SP14%20CS188%20Lecture%202%20--%20Uninformed%20Search.pptx). *UC Berkeley CS188 Intro to AI*. Retrieved 12 November 2019.

---