Introduction to Sequential Decision-Making: Using Relational Representations

Siddharth Srivastava, Ph.D.
Assistant Professor
Arizona State University

Ira A. Fulton Schools of Engineering
Arizona State University

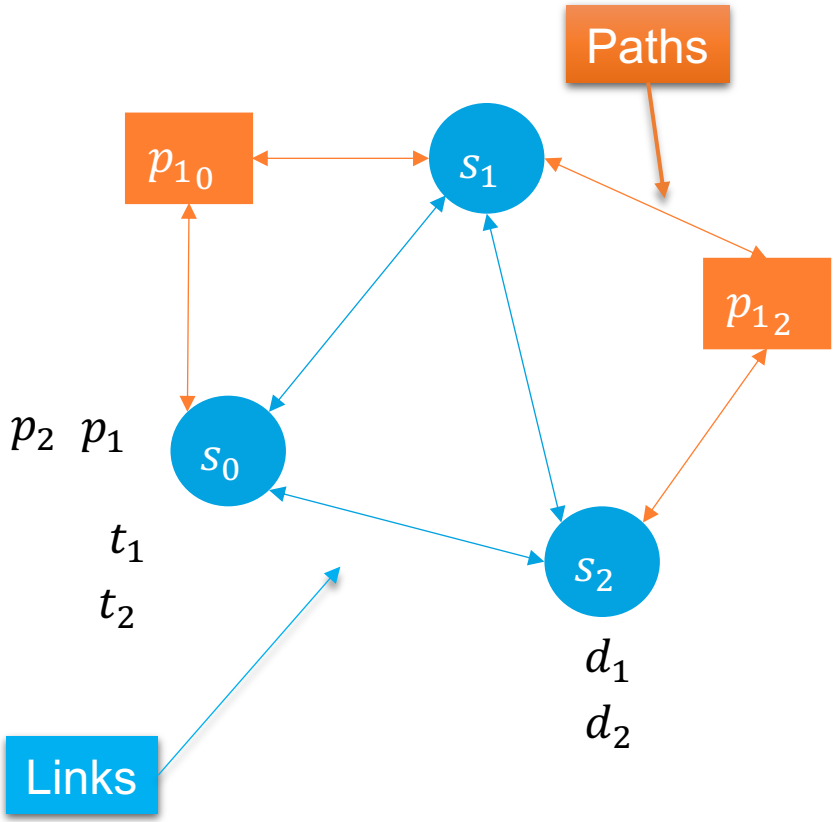
Recall: Relational Representations

- We have discussed the limitations of naive search representations and how to overcome them using relational representations

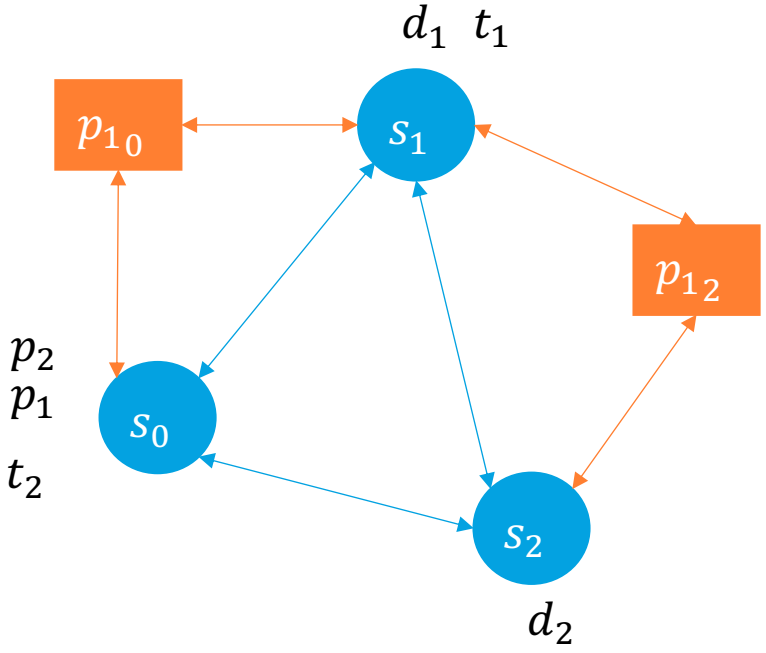
- Now, we will address how to use relational representations for fast, scalable planning

Example Problem

Initial State



Goal State



#states: 3 possible locations for trucks, packages
7 possible locations for drivers

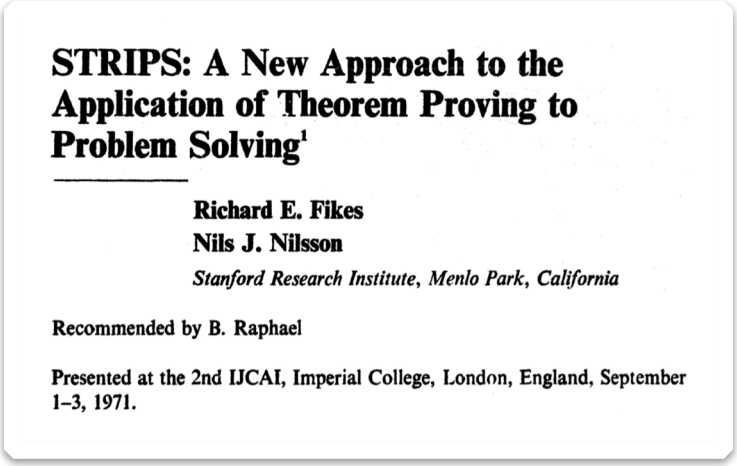
$$3^4 \times 7^2 = 3969$$

SHAKEY and STRIPS

Original idea behind the STRIPS system language and system

- Express states using a relational representation
- Original approach utilized first-order logic (FOL)
- Use a theorem prover to select actions that may be useful

The problem representation outlasted the algorithmic approach



STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving[1]

Richard E. Fikes
Nils J. Nilsson
Stanford Research Institute, Menlo Park, California

Recommended by B. Raphael

Presented at the 2nd IJCAI, Imperial College, London, England, September 1–3, 1971.

***push*(*k*, *m*, *n*)**

Precondition: **ATR(*m*) ∧ AT(*k*, *m*)**

delete list **ATR(*m*); AT(*k*, *m*)**

add list **ATR(*n*); AT(*k*, *n*)** .

Modern Planning Domain Descriptions

Planning Domain Definition Language (PDDL)

Domain:

- Vocabulary
  - Constants, Predicates, Functions, Types
- Action specifications:
  - Precondition: First-order logic (FOL) formula using the vocabulary
  - Effects: Facts that are added or removed by the action

```
(:predicates   (obj ?obj)
    (truck ?truck)
    (location ?loc)
    (driver ?d)
    (at ?obj ?loc)
    (in ?obj1 ?obj2)
    (driving ?d ?v)
    (link ?x ?y) (path ?x ?y)
    (empty ?v)
)
```

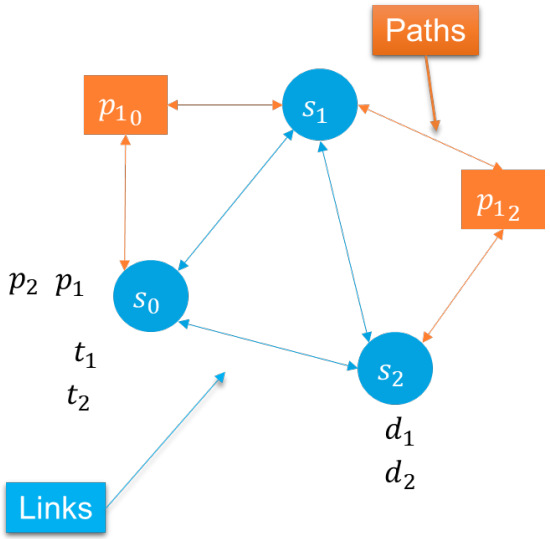
PDDL: Action Descriptions

Let's define the load action as follows

- "You can load trucks with objects at their location. Upon loading, the object is no longer at its location – it is in the truck."

```
(:predicates   (obj ?obj)
    (truck ?truck)
    (location ?loc)
    (driver ?d)
    (at ?obj ?loc)
    (in ?obj1 ?obj)
    (driving ?d ?v)
    (link ?x ?y) (path ?x ?y)
    (empty ?v)
)
```

```
( :action LOAD-truck
   :parameters
    (?obj ?truck ?loc)
   :precondition
    (and (OBJ ?obj) (truck ?truck) (location ?loc)
    (at ?truck ?loc) (at ?obj ?loc))
   :effect
    (and (not (at ?obj ?loc)) (in ?obj ?truck)))
```




```
(:predicates   (obj ?obj)
    (truck ?truck)
    (location ?loc)
    (driver ?d)
    (at ?obj ?loc)
    (in ?obj1 ?obj)
    (driving ?d ?v)
    (link ?x ?y) (path ?x ?y)
    (empty ?v)
)
```

```
(:action DRIVE-truck
   :parameters
    (?truck
     ?loc-from
     ?loc-to
     ?driver)
   :precondition
    (and (truck ?truck) (location ?loc-from)(location ?loc-to)
    (driver ?driver)
    (at ?truck ?loc-from)
    (driving ?driver ?truck) (link ?loc-from ?loc-to))
   :effect
    (and (not (at ?truck ?loc-from)) (in ?truck ?loc-to)))
```

```
( :action BOARD-truck
    :parameters
     (?driver
      ?truck
      ?loc)
    :precondition
     (and (driver ?driver) (truck ?truck) (location ?loc)
     (at ?truck ?loc) (at ?driver ?loc) (empty ?truck)
    :effect
     (and (not (at ?driver ?loc)) (driving ?driver ?truck) (not (empty ?truck))))

( :action DISEMBARK-truck
    :parameters
     (?driver
      ?truck
      ?loc)
    :precondition
     (and (driver ?driver) (truck ?truck) (location ?loc)
          (at ?truck ?loc) (driving ?driver ?truck)
    :effect
     (and (not (driving ?driver ?truck)) (at ?driver ?loc) (empty ?truck)))
    .
    .
    .
( :action WALK
    :parameters
     (?driver
      ?loc-from
      ?loc-to)
    :precondition
     (and (driver ?driver) (location ?loc-from) (location ?loc-to)
       (at ?driver ?loc-from) (path ?loc-from ?loc-to))
    :effect
     (and (not (at ?driver ?loc-from)) (at ?driver ?loc-to)))
```

Summary: Planning Domain Definition

Planning Domain:

- Vocabulary
  - Constants, Predicates, Functions, Types
- Action specifications
  - Precondition: FOL formula using the vocabulary
  - Effects: Facts that are added or removed by the action

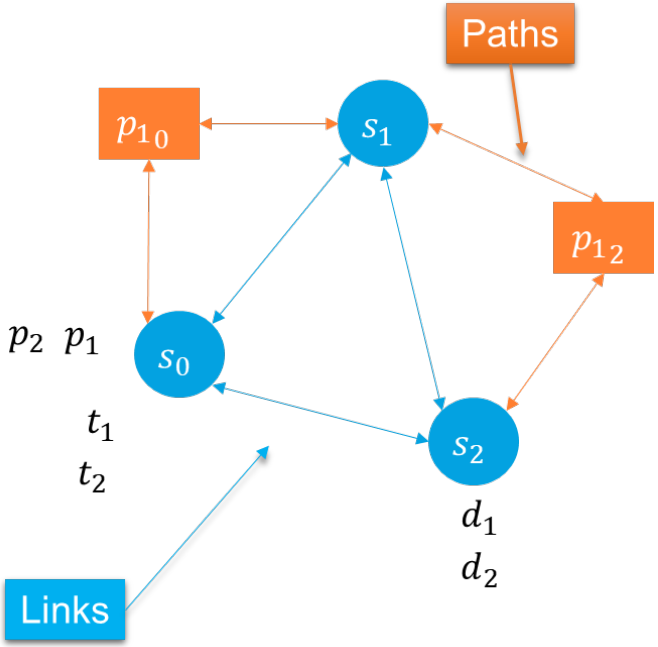
Planning Problem:

- Model of the initial state (recall FOL semantics)
  - Universe: list of objects of each type
  - Interpretation: settings of each predicate, function
- Goal formula

Defining a Planning Problem

Problem:

- Model of the initial state
- Universe: list of objects of each type
- Interpretation: settings of each predicate, function

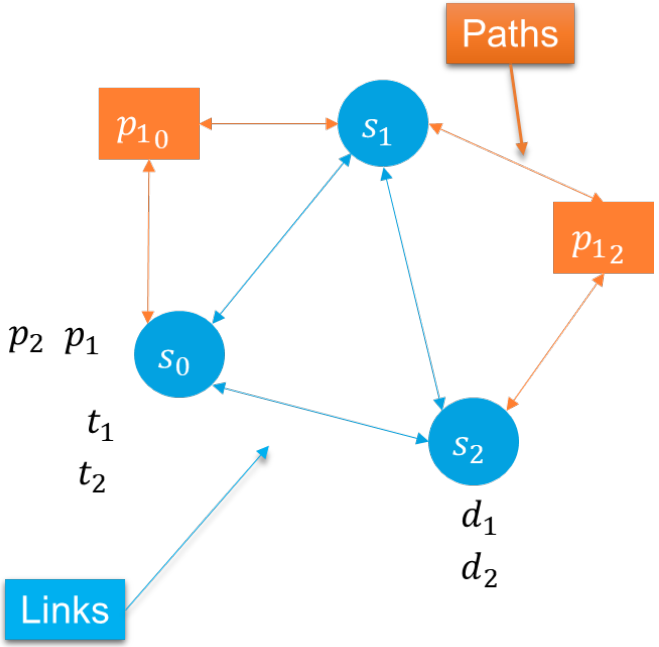



```
(:init
(at driver1 s2)
(DRIVER driver1)
(at driver2 s2)
(DRIVER driver2)
(at truck1 s0)
(empty truck1)
(TRUCK truck1)
(at truck2 s0)
(empty truck2)
(TRUCK truck2)
(at package1 s0)
(OBJ package1)
(at package2 s0)
(OBJ package2)
(LOCATION s0)
(LOCATION s1)
(LOCATION s2)
(LOCATION p1-0)
(LOCATION p1-2)
```

Defining a Planning Problem

Problem:

- Model of the initial state
- Universe: list of objects of each type
- Interpretation: settings of each predicate, function




```
(:init
(at driver1 s2)
(DRIVER driver1)
(at driver2 s2)
(DRIVER driver2)
(at truck1 s0)
(empty truck1)
(TRUCK truck1)
(at truck2 s0)
(empty truck2)
(TRUCK truck2)
(at package1 s0)
(OBJ package1)
(at package2 s0)
(OBJ package2)
(LOCATION s0)
(LOCATION s1)
(LOCATION s2)
(LOCATION p1-0)
(LOCATION p1-2)
```

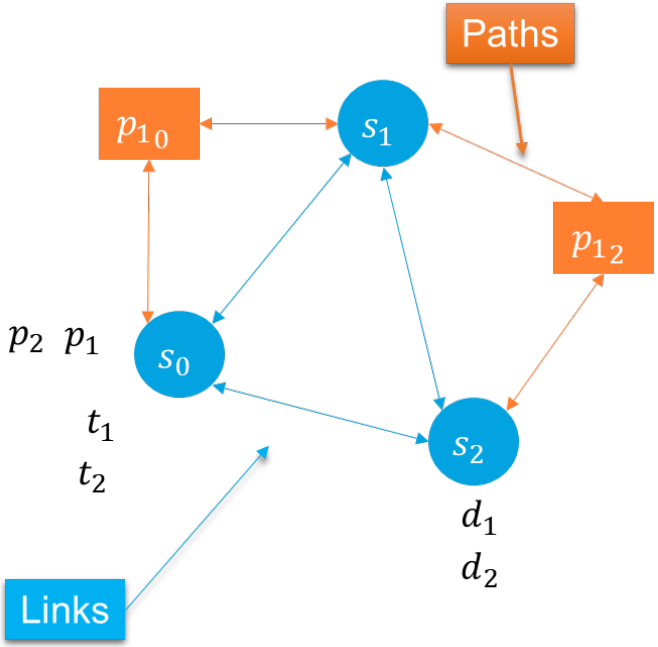
Closed World Assumption

No need to mention negated literals in the initial state.
If a literal is not mentioned, it is assumed to be false.

Defining a Planning Problem

Problem:

- Model of the initial state
- Universe: list of objects of each type
- Interpretation: settings of each predicate, function



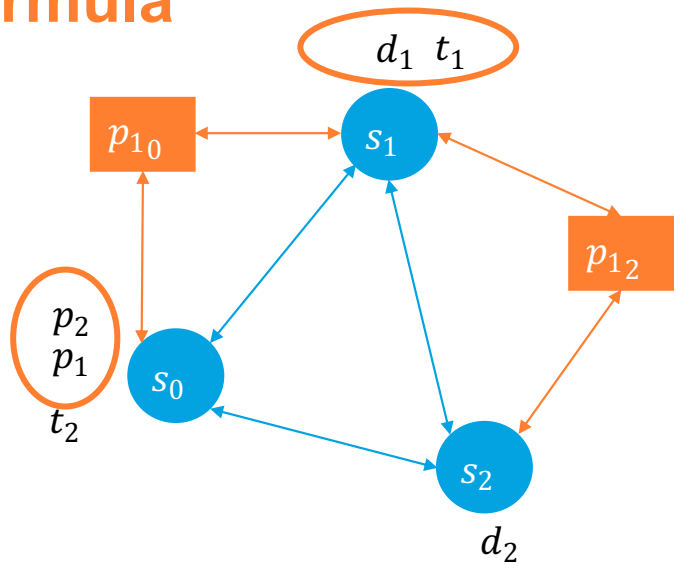

```
(:init
(at driver1 s2)
(DRIVER driver1)
(at driver2 s2)
(DRIVER driver2)
(at truck1 s0)
(empty truck1)
(TRUCK truck1)
(at truck2 s0)
(empty truck2)
(TRUCK truck2)
(at package1 s0)
(OBJ package1)
(at package2 s0)
(OBJ package2)
(LOCATION s0)
(LOCATION s1)
(LOCATION s2)
(LOCATION p1-0)
(LOCATION p1-2)
(path s1 p1-0)
(path p1-0 s1)
(path s0 p1-0)
(path p1-0 s0)
(path s1 p1-2)
(path p1-2 s1)
(path s2 p1-2)
(path p1-2 s2)
(link s0 s1)
(link s1 s0)
(link s0 s2)
(link s2 s0)
(link s2 s1)
(link s1 s2)
```

Defining a Planning Problem

Problem:

- Model of the initial state
- Universe: list of objects of each type
- Interpretation: settings of each predicate, function

Goal formula

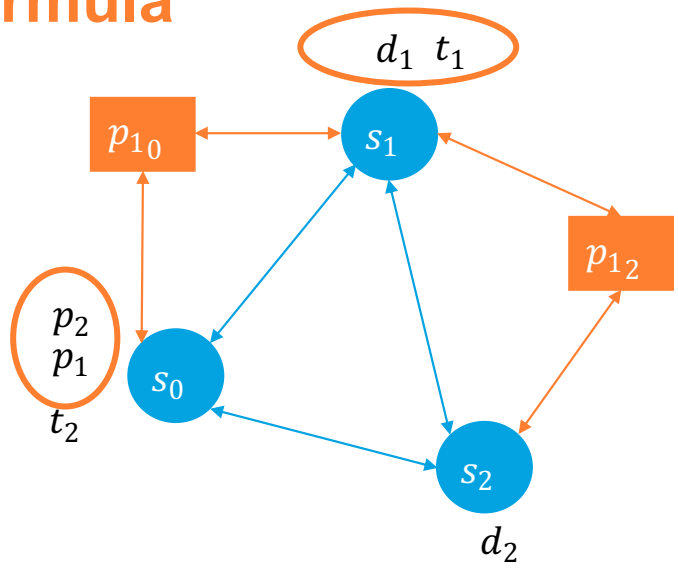


Defining a Planning Problem

Problem:

- Model of the initial state
- Universe: list of objects of each type
- Interpretation: settings of each predicate, function

Goal formula




```
(:goal (and
(at driver1 s1)
(at truck1 s1)
(at package1 s2)
(at package2 s2)
))
```

Advantages of this Representational Framework

- **A single domain can be used to define and solve infinitely many problems**
- **Changes in the agent's objectives require changes only in the problem definition (not in the domain definition)**
  - E.g., different logistics companies, different package lists, different countries
- **You can try solving such problems using open-source planners such as Fast-Forward or Fast-Downward**
  - These planners can solve problems with millions of states in the logistics domain in seconds!
- **Next: How do they do it?**