CSE 579: Knowledge Representation & Reasoning

# Module 5:
# Reasoning about Actions

Ali Altunkaya
Spring 2023

# Outline

1. Introduction to Reasoning about Actions
2. Simple Transition Systems in ASP
3. More complex Transition Systems in ASP
4. Expressive Possibilities
    1. Non-deterministic Actions
    2. Interaction between Concurrent Actions
    3. Non-Inertial Fluents

# 1. Introduction

In Module 4, we learned how to solve Combinatorial Search problems using ASP theory and Clingo:

- Graph coloring
- Vertex cover
- Hamiltonian cycle
- Sudoku, etc...

In this Module 5, we will tackle with different kinds of problems. Mostly in Robotics, we have an initial state and some available actions and then we want to reach a goal or final state, using a sequence of actions.

- Monkey-Bananas problem, assume Monkey is a robot
- Blocks World problem, etc...

# 1. Introduction

Until now, we are only interested in atoms (facts) which has a state:

- Each atom can be either True or False.

Goal: Developing appropriate formalism for describing the properties of actions.

**What is action?**

- Anything (an atom or function) that can be executed and change the state of the world (another atom).
- Actions has no state (True/False) in this formalism.
- Actions are executed and change the state of other atoms.

**Why we need actions?**

- To solve complex problems, we need a sequence of actions to reach the goal.
- Since there is sequence, there will be time parameter to represent the order of actions within the sequence:
  - T1, T2, T3 etc. 0, 1, 2, 3, 4, …..... m

# 1. Introduction

**Goal:** Developing appropriate formalism for describing the properties of actions.

There is no new keyword or tag for actions, but we will use the following formalism.

- Actions are defined in the body, and its effect is defined in the head.
- Since time is important for sequence of actions. We will add time as the last parameter to all fluents and actions.
- Head ← Body
- (Effect on fluent) ← (Action)
- p(t, 1) ← a(0)

# 1. Introduction

**Fluents:** Fluent is like atom. Anything (an atom or function), its value changes over time, depending on the state of the world. Fluent has two characteristics:

- Its value changes over time, so it always has time parameter.
  - time1: p is True  time2: p is False  time3: p is True  …….
- The value of fluents can be either:
  - True/False: monkeyHasBananas : T/F
  - Multi-valued: location(monkey): L1, L2, L3, ….

**Commonsense law of Inertia:** By default, the values of fluents remain unchanged, after executing actions.

- Because it will be infeasible to enumerate all things that don't changed.

How to formalize the commonsense law of inertia in ASP, and how to represent inertial fluents. This is known as **The Frame Problem**.

# 2. Simple Transition System

## Transition Systems:

- We use transitions systems to develop appropriate formalism for describing actions.
- Transition systems can be represented by directed graphs where
  - Vertices corresponds to States
  - Edges corresponds to Actions

**ASP solution to Frame Problem:** using defaults

```
{p(t,1)} :- p(t,0).
{p(f,1)} :- p(f,0).
:- not 1{p(t,1); p(f,1)}1.
```

# 3. Ramification and Query types

**Ramification problem:** How to describe indirect effects of an action?

- Direct effects are obvious.
- There might be many indirect effects, but we don't need to write all indirect effects of an action, most will be redundant.
- If preconditions conflict, then we don't need those redundant rules for indirect effects.

**Query Types:**

**1-) Planning Query:** Given initial state and final state of a fluent, find (predict) the shortest sequence of actions. Most obvious case.

**2-) Prediction Query:** Given initial state of a fluent and a sequence of actions, can we predict the final state of a fluent?

**3-) Postdiction Query:** Given a sequence of actions and final state of a fluent, can we predict the initial state of a fluent?

# 4. Expressive Possibilities

**1-) Non-Deterministic Actions:** Until now, we have used deterministic actions.

- Deterministic actions have only one possible effect. e.g.:
  - loc(jack,L,T+1) :- go(L,T)
- Non-Deterministic actions may have more than one possible results/effects. We use choice rule to formalize non-deterministic actions. e.g.:
  - {loc(car,L,T+1)} :- go(L,T), loc(car,L1,T), loc(jack,L1,T), T=0..m-1.

**2-) Concurrent Actions:** can be formalized using the same time as below.

- onTable(f,T) :- level(leftEnd,H,T), level(rightEnd,H1,T), H!=H1

**2-) Non-Inertial Fluent:** By default, fluents are inertial according to the Commonsense Law of Inertia.

- Inertial fluents: It stays in the same state, unless there is no effect.
- Non-Inertial fluents: The state of a fluent change by time without any effect. For example pendulum:
  - {left(T+1)} :- right(T), T=0..m-1.
  - {right(T+1)} :- left(T), T=0..m-1.

Thanks

&

Questions