



| Lecture Notes No. 7 | | | |
|----------------------|---|-----------------------|-----------------------------|
| Topic: | Situation Calculus | Week No. | 10 |
| Course Code: | CSST101 | Term: | 1 st Semester |
| Course Title: | Advance Knowledge Representation and Reasoning | Academic Year: | 2025-2026 |
| Student Name | | Section | |
| Due date | | Points | |

Learning Outcomes

By the end of this lesson, you should be able to:

1. Explain the concept of situation calculus and its components.
2. Model actions, fluents, and situations formally.
3. Encode actions and state changes in Python.
4. Apply reasoning to predict outcomes of sequences of actions.

1. What is Situation Calculus?

Situation calculus is a formalism used to represent and reason about **dynamic worlds**, where actions cause changes in the environment.

Example:

- “John moves the box from room A to room B.”
- The world changes depending on the action taken.
- We represent this using **situations** (states), **actions**, and **fluents** (properties that may change).

Definition:

Situation calculus describes how **actions transform situations** and how the truth of properties (fluents) depends on the situation.

2. Key Concepts

| Component | Description | Example |
|--------------------|---|------------------------------|
| Action | Something that changes the world | move_box(John, A, B) |
| Fluent | A property that may change over time | box_at(Box, Room) |
| Situation | A snapshot of the world | S0 = initial situation |
| Successor Function | Describes the new situation after an action | do(move_box(John, A, B), S0) |

Key Idea:

- Each action leads to a new situation.
- Reasoning involves predicting the effects of sequences of actions.



3. Representation in Python

We can encode actions and situations using **classes and functions**.

Example:

```
class Situation:  
    def __init__(self, box_location):  
        self.box_location = box_location  
  
    def move_box(situation, new_location):  
        return Situation(new_location)  
  
# Initial situation  
S0 = Situation("Room A")  
S1 = move_box(S0, "Room B")  
print(S1.box_location) # Output: Room B
```

Exercise:

- Add multiple boxes and actions.
- Predict final locations after sequences of actions.

4. Modeling Actions and Fluents

1. Fluent Representation:

- at(object, location, situation)

2. Action Representation:

- move(object, from, to)

3. Situation Update:

- do(action, situation) returns a new situation reflecting changes

Example Reasoning:

- If box is at A in S0 and John moves it to B, then box is at B in S1.

5. Exercises

1. Encode a robot that can move between rooms and pick up/drop objects.
2. Model the world state as situations after a sequence of actions.
3. Query: “Where is the box after John moves it from A → B → C?”



6. Applications in AI

| Field | Example of Use |
|--------------------|--|
| Robotics | Planning sequences of actions for robots |
| Automated Planning | Predict outcomes of actions in dynamic systems |
| Games | Modeling character actions and effects |
| Logistics | Tracking object movement in warehouses |

7. Lab Activity (Python)

Goal: Encode a world with objects and locations, then simulate actions.

Sample Tasks:

- Define classes for actions and situations.
- Write functions for `do(action, situation)`.
- Observe changes in fluents as actions are executed.

Challenge:

- Add rules for constraints, e.g., "a robot cannot move if the path is blocked."

8. Reflection and Discussion

1. How does situation calculus handle changes in the world compared to classical logic?
2. Can you think of a real-life scenario where predicting sequences of actions is important?
3. How would you extend this model to multiple agents interacting?

9. Summary

- Situation calculus models **dynamic worlds** with actions, fluents, and situations.
- Actions lead to new situations, allowing reasoning about state changes.
- Python can be used to simulate and query situations.
- Useful in robotics, planning, games, and logistics.

Self-Check

1. What are the main components of situation calculus?
2. How does the successor function work?
3. How do fluents differ from actions?
4. How can Python help simulate situation calculus reasoning?