Republic of the Philippines
Laguna State Polytechnic University
Province of Laguna

Lecture Notes No. 8

Topic:	**Event Calculus & Temporal Reasoning**	**Week No.**	11-12
Course Code:	**CSST101**	**Term:**	1st 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 concepts of event calculus and temporal reasoning.
2. Model events, fluents, and time points in dynamic domains.
3. Implement happens and holdsAt predicates in Python.
4. Query temporal relationships to reason about sequences of events.

1. What is Event Calculus?

Event calculus is a formalism used to **represent and reason about events and their effects over time.** Unlike situation calculus, which focuses on discrete situations, event calculus emphasizes **time and durations**.

Example:

- "John turns on the light at 9:00 AM."
- "The light remains on until 5:00 PM."
- Event calculus can answer: "Is the light on at 3:00 PM?"

Definition:

Event calculus allows us to model **fluents (properties)** that are true over certain intervals of time and to reason about **the effects of events**.

2. Key Concepts

Component	**Description**	**Example**
Event	Something that happens at a specific time	turn_on(light, 9)
Fluent	A property that can change over time	light_on(light)
Time Point	A specific moment or instant	t = 9
Predicate happens(e, t)	Event e occurs at time t	happens(turn_on(light), 9)
Predicate holdsAt(f, t)	Fluent f is true at time t	holdsAt(light_on(light), 10)

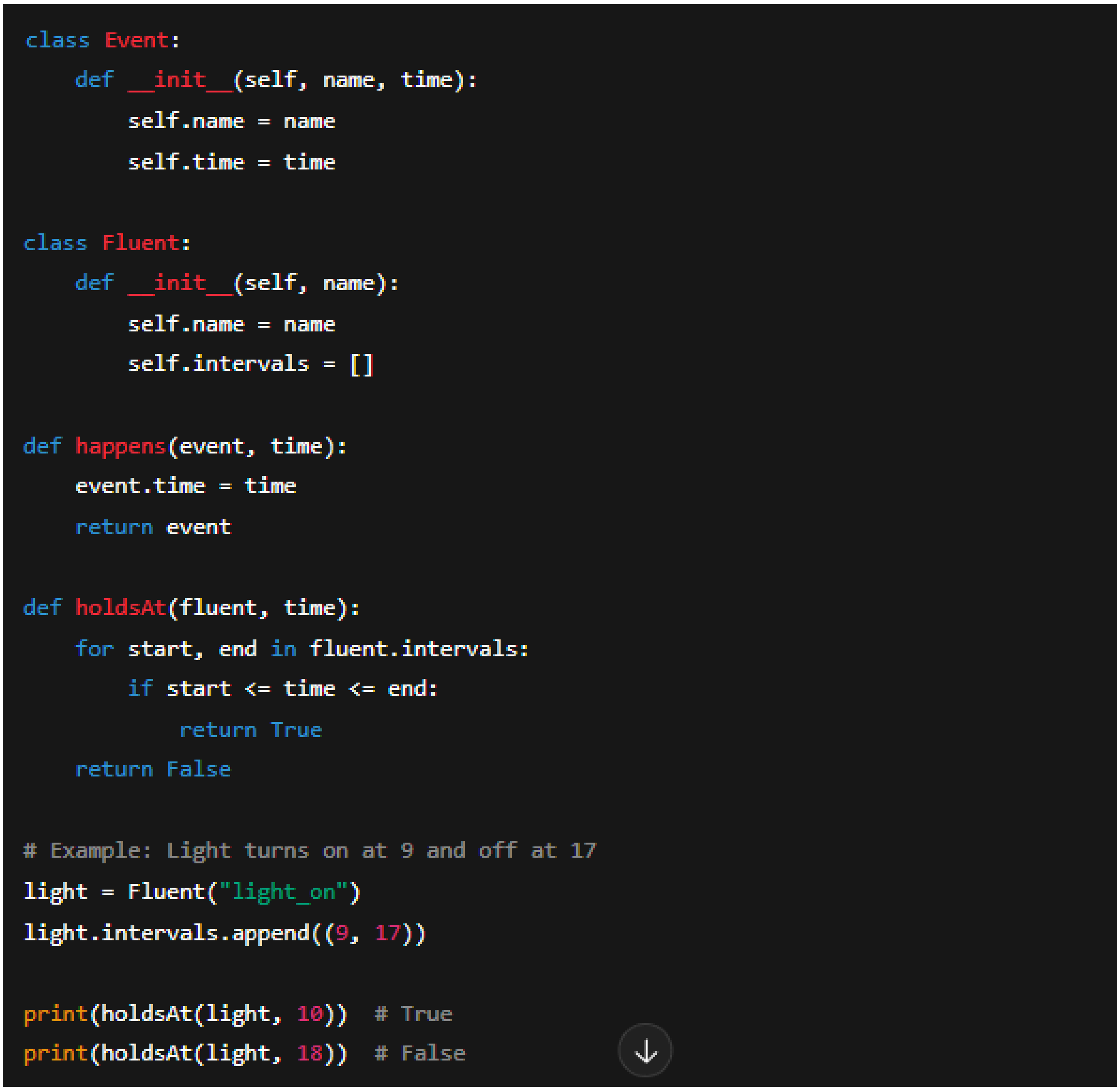
Key Idea:

- Fluents can **hold for intervals** between events.
- Events initiate or terminate fluents.

3. Representing in Python

We can encode **events, fluents, and their temporal relationships** using Python.

Example:



```python
class Event:
    def __init__(self, name, time):
        self.name = name
        self.time = time

class Fluent:
    def __init__(self, name):
        self.name = name
        self.intervals = []

def happens(event, time):
    event.time = time
    return event

def holdsAt(fluent, time):
    for start, end in fluent.intervals:
        if start <= time <= end:
            return True
    return False

# Example: Light turns on at 9 and off at 17
light = Fluent("light_on")
light.intervals.append((9, 17))

print(holdsAt(light, 10))  # True
print(holdsAt(light, 18))  # False
```

Exercise:

- Add events for turning devices on/off.
- Query the state of fluents at different times.

4. Temporal Queries

1. **What happens before/after a time?**
2. **Does a fluent hold at a given time?**
3. **What is the next event affecting a fluent?**

Example Query:

- "Was the light on at 12 PM?" → Yes, because the event turn_on occurred at 9:00 and no terminating event has occurred.

5. Applications in AI

Field	**Example of Use**
Robotics	Scheduling actions and reasoning about durations
Process Monitoring	Detecting system states over time
Intelligent Agents	Predicting outcomes based on events
Smart Homes	Managing devices and automating behaviors

6. Lab Activity (Python)

Goal: Implement temporal reasoning with happens and holdsAt.

Sample Tasks:

- Encode a timeline of events for a smart home.
- Implement functions to check if fluents hold at certain times.
- Observe changes when new events are added.

Challenge:

- Add overlapping events and see how they affect fluents.

7. Reflection and Discussion

1. How is event calculus different from situation calculus?
2. Can you think of a real-world scenario where temporal reasoning is essential?
3. How does reasoning over time help AI make better decisions?

8. Summary

- Event calculus models **events, fluents, and time points**.
- Fluents are **initiated or terminated** by events.
- Temporal queries allow reasoning about the **state of the world at any time**.
- Useful for AI applications in robotics, monitoring, smart systems, and scheduling.

Self-Check

1. What is the difference between happens and holdsAt?
2. How does event calculus handle overlapping events?
3. How can Python be used to simulate temporal reasoning?
4. Why is temporal reasoning important for real-world AI systems?