(Qualitative) Reasoning over Time

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Prof. Ing. Federico Chesani

DISI

Department of Informatics – Science and Engineering

Motivation

- We live in a temporal dimension: our environment change as time goes by
- Categories, Substances, Objects and their properties provide a static perception
- (depending on the application) We might need to take into account also a dynamic dimension, related to the passing of time
  - about the execution of actions
  - about events happening
  - about how such events (will) change the environment

Aims:

- reason about the past
- predict future (planning)

Goal of this lesson

Qualitative Time reasoning

- Prop. logic for representing dynamics, Effect Axioms
- Frame problem, Frame axioms, representational frame problem

- Event Calculus
- Allen's Temporal Logic
- CEP as a "meta-example"

Disclaimer and Further reading

Reading:

- AIMA, chapter 7, Section 7.7
- AIMA, chapter 10, Section10.3

Further reading:

- Kowalski & Sergot paper on EC, 1986
- Shanahan 1999 paper with its version of EC
- J. F. Allen, Maintaining knowledge about temporal intervals, Communications of the ACM, 1983

Propositional Logic for representing dynamics, and the Frame problem

How to represent the current state of the world?

Wittgenstein in a 1922 essay faced the problem of representing the belief of an agent w.r.t. the world.

The idea:

- use propositional logic, and propositions, to represent an agent's beliefs about the world
- a current state of the world would be represented as a set of propositions
  - in the set it would appear only "true" propositions, thus representing what the agent believes to be true

- different sets, put in some order, would represent the world evolution perceived by an agent
  - sets and propositions in the sets would be marked with numerical apex to establish the evolution order

How to represent the current state of the world?

Example:

A child has a bow and an arrow. She can shoot the arrow, throw the bow, but she can also run, hide, and crouch. Our agent ***observes*** *the child, and it believes/knows that:*

$KB^0 = \{hasBow^0, hasArrow^0\}$

The child shoots the arrow, our agent ***observes*** *the child and it believes/knows that:*

$KB^1 = \{hasBow^0, hasArrow^0, hasBow^1, \neg hasArrow^1\}$

How to represent the dynamic of the world?

Example:

A child has a bow and an arrow. She can shoot the arrow, throw the bow, but she can also run, hide, and crouch. Our agent ***observes*** *the child, and it believes/knows that:*

$KB^0 = \{hasBow^0, hasArrow^0\}$

The child shoots the arrow, our agent ***observes*** *the child and it believes/knows that:*

$KB^1 = \{hasBow^0, hasArrow^0, hasBow^1, \neg hasArrow^1\}$

We are interested to focus on the dynamic, or rather how the world evolves from KB^0 to KB^1

- What happened in between?
- There is some idea of action, whose execution changes the world…

How to represent the dynamic of the world?

- There is a notion of **state** that captures the world at a certain instant.
- The evolution of the world is given as a sequence of states…
- States are **countable** …

- There are **actions** …
- Actions affect how each state evolves to the next one…
- We would describe such effect in terms of **effect axioms**

Effect Axioms

Example

A child has a bow and an arrow. She can shoot the arrow, throw the bow, but she can also run, hide, and crouch.

The child shoots the arrow…

What does it mean "shooting an arrow"?

$$shoot^{t} \Rightarrow (hasArrow^{t} \Leftrightarrow \neg hasArrow^{(t+1)})$$

$$KB^{0} = \{hasArrow^{0}\}$$

$$KB^{1} = \{hasArrow^{0}, \neg hasArrow^{1}\}$$

Effect Axioms

Example – complete

A child has a bow and an arrow. She can shoot the arrow, throw the bow, but she can also run, hide, and crouch.

The child shoots the arrow…

What does it mean "shooting an arrow"?

$shoot^t \Longrightarrow (hasArrow^t \Leftrightarrow \neg hasArrow^{(t+1)})$

$KB^0 = \{hasBow^0, hasArrow^0\}$

$KB^1 = \{hasBow^0 , hasArrow^0, \neg hasArrow^1\}$

WAIT! That is different from:

$KB^1 = \{hasBow^0, hasArrow^0, hasBow^1, \neg hasArrow^1\}$

Effect Axioms

Example – complete

A child has a bow and an arrow. She can shoot the arrow, throw the bow, but she can also run, hide, and crouch.

The child shoots the arrow…

What does it mean "shooting an arrow"?

$$shoot^{t} \Rightarrow (hasArrow^{t} \Leftrightarrow \neg hasArrow^{(t+1)})$$

$$KB^{0} = \{hasBow^{0}, hasArrow^{0}\}$$

$$KB^{1} = \{hasBow^{0}, hasArrow^{0}, \neg hasArrow^{1}\}$$

Does the child still have the bow after the shoot? In other word, $hasBow^{1}$ is TRUE or FALSE?

The Frame problem

- The effect axioms fail to state what remains unchanged as the result of an action.
- It is named "**Frame problem**" because it refers to the fact that the "background" remains unchanged, while the "foreground" is subjected to the effects…
- … but the "effect axiom" tell us something about the "foreground"... what about the "background", alias frame?

A child has a bow and an arrow. She can shoot the arrow, throw the bow, but she can also run, hide, and crouch.

The child shoots the arrow…

$shoot^t \Rightarrow (hasArrow^t \Leftrightarrow \neg hasArrow^{(t+1)})$

$KB^0 = \{hasBow^0, hasArrow^0\}$

$KB^1 = \{hasBow^0 , hasArrow^0, \neg hasArrow^1\}$

A solution to the Frame problem: frame axioms

- A solution would be the **frame axioms**
- For each proposition that is not affected by the action, we will state that it is unaffected

A child has a bow and an arrow. She can shoot the arrow, throw the bow, but she can also run, hide, and crouch.

The child shoots the arrow…

$shoot^t \Rightarrow (hasArrow^t \Leftrightarrow \neg hasArrow^{(t+1)})$

$shoot^t \Rightarrow (hasBow^t \Leftrightarrow hasBow^{(t+1)})$

$KB^0 = \{hasBow^0, hasArrow^0\}$

$KB^1 = \{hasBow^0 , hasArrow^0, hasBow^1 , \neg hasArrow^1\}$

A solution to the Frame problem: frame axioms

Problems:

- If we have *m* actions and *n* propositions, the set of frame axioms will be $O(mn)$… representational frame problem
- Reasoning about *t* steps ahead will have a temporal complexity of $O(nt)$… inferential frame problem
- Locality principle mitigate the problems above, does not eliminate them

Objection: do we really need to assert frame axioms?

- we could use logics for stating a "sort of super axiom" that say that everything that is untouched remain the same
- it is not possible to do it in FOL, we can do it in high order logic

The Situation Calculus

The Successor State Axioms

- Effects and Frames axioms focuses on the actions
- A solution consists on changing the viewpoint: focus on the propositions describing the world, that we will be named fluents

Ray Reiter (1991): Successor State Axioms

Each state is described by a set of fluents F. Then, we define the following axioms:

$F^{(t+1)} \Leftrightarrow ActionCausesF^{t} \lor (F^{t} \land \neg ActionCausesNotF^{t})$

Example:

$hasBow^{(t+1)} \Leftrightarrow pickUpBow^{t} \lor (hasBow^{t} \land \neg throwBow^{t})$

The Situation Calculus

Proposed by John McCarthy (1963); Ray Reiter (1991)

It aims at Planning as first-order logical deduction

- The initial state is called a *situation*
- If *a* is an action and *s* a situation, then *Result(s, a)* is a situation
- A function/relation that can vary from one situation to the next is called a *fluent*: *At(x, l, s)*

- Introduces preconditions of an action (as in planning)
- Action's preconditions are defined by *possibility axioms:*

$$\phi(s) \implies Poss(a,s)$$

  - E.g.: hasArrow(s) $\wedge$ hasBow(s) $\implies$ Poss(shoot, s)

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

The Situation Calculus

- Adopt the Successor state axioms, but adapted with the possibility notion:

  $F^{(t+1)} \Leftrightarrow \text{ActionCausesF}^t \lor (F^t \land \neg\text{ActionCausesNotF}^t)$
  $Poss(a, s) \Rightarrow (F(Result(a,s)) \Leftrightarrow a = ActionCausesF \lor (F(s) \land a \neq ActionCausesNotF)$
  $Poss(shoot, s) \Rightarrow (\neg hasArrow(Result(a,s)) \Leftrightarrow a = shoot \lor \neg hasArrow(s) \land a \neq reload)$
- To avoid non-determinism and/or conflicts/incoherencies, it has a "unique action" axiom: only one action can be executed in a situation
- The goal is defined as a conjunction of fluents
- The solution is a situation, i.e. a sequence of actions, that satisfies the goal.
- Deduction is used to compute the right action sequence.

The Situation Calculus – problems

- McCarthy's and Green's formulation maps fluents into *predicates*; they do not allow to assert about the truthness of a fluent in a situation
- Actions are discrete, instantaneous, and happen one at a time
- Two actions cannot happen at the same time
- The Situation Calculus defines:
  - What is true before the action…
  - What is true after the action…
  - … but say nothing on what is true during the action

The Event Calculus

Event Calculus

Proposed by Marek Sergot and Robert Kowalski, 1986

- Based on *points of time*
- Reifies both fluents and events into terms: HoldsAt(fluent, situation/action)
- **Fluents** are properties whose truthness value changes over time

- Advantages of a second-order formula, yet still first order
- Allows to link multiple different events to the same state property (named fluent)
- State property changes can depend also from other states
- Allows to reason on meta-events of state property changes (clip and de-clip meta-events)

Event Calculus

The formulation comprises an ontology and two distinct set of axioms:

1. Event calculus "ontology" (fixed)
2. Domain-independent axioms (fixed)
3. Domain-dependent axioms (application dependent)

Event Calculus – EC Ontology

- **HoldsAt**(F, T): The fluent F holds at time T
- **Happens**(E, T): event E (i.e., the fact that an action has been executed) happened at time T

- **Initiates**(E, F, T): event E causes fluent F to hold at time T (used in domain-dependent axioms…)
- **Terminates**(E, F, T): event E causes fluent F to cease to hold at time T (used in domain-dependent axioms…)

- **Clipped**(T_1, F, T): Fluent F has been made false between T_1 and T(used in domain-independent axioms), $T_1 < T$
- **Initially**(F) : fluent F holds at time 0

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Event Calculus – Domain-independent Axioms

Two axioms define when a fluent is true:

- $HoldsAt(F, T) \Leftarrow Happens(E, T_1) \land Initiates(E, F, T_1) \land (T_1 < T) \land \neg Clipped(T_1, F, T)$

- $HoldsAt(F, T) \Leftarrow Initially(F) \land \neg Clipped(0, F, T)$

An axiom defines the clipping of a fluent:

- $Clipped(T_1, F, T_2) \Leftarrow Happens(E, T) \land (T_1 < T < T_2) \land Terminates(E, F, T)$

Event Calculus – Domain-dependent Axioms

A collection of axioms of type Initiates(…)/Terminates(…), and Initially(…)

- Initially(F): the fluent F holds at the beginning
- Initiates(Ev, F, T): the happening of event Ev at time T makes F to hold; it can be extended with many (pre-)conditions
- Terminates(Ev, F, T): the happening of event Ev at time T makes F to not hold anymore.

Event Calculus – Example

Example: we have a single button in a room: the pressing of the button switch on or off the light.

Fluents:

- light_on, light_off

  why not light(on) vs light(off) ??? it's fine as well…

Events:

- push_button.

Event Calculus – Example

Example: we have a single button in a room: the pressing of the button switch on or off the light.

Domain-dependent axioms, initial state of the world:

- Initially(light_off).

Effects of the "push_button" event on the fluent **light_on**:

- Initiates(push_button, light_on, T) $\Leftarrow$ HoldsAt(light_off, T).
- Terminates(push_button, light_on, T) $\Leftarrow$ HoldsAt(light_on, T).

Effects of the "push_button" event on the fluent **light_off**:

- Initiates(push_button, light_off, T) $\Leftarrow$ HoldsAt(light_on, T).
- Terminates(push_button, light_off, T) $\Leftarrow$ HoldsAt(light_off, T).

Event Calculus – Example

Given a set of events:

- Happens(push_button, 3)
- Happens(push_button, 5)
- Happens(push_button, 6)
- Happens(push_button, 8)
- Happens(push_button, 9)

Is the light on?

Event Calculus allows to answer the queries about HoldsAt predicates very easily

Event Calculus is very important… why?

- It allows to represent the state of a system in logical terms, in particular FOL
- It allows to represent and reason on how a system evolves as a consequence of happening events
- It is an easier formalism (w.r.t. other solutions)
- It is declarative/logic based

Applications:

- Monitoring of systems
- Simulation of system's evolutions
- Practically, adopted as a principled framework for representing system evolutions and reactiveness

Event Calculus – few limits…

- Easily implemented in Prolog…
- …but (roughly) not safe if fluents/events contain variables, due to the negation in front of the clipping test, that is implemented in Prolog through NAF

Event Calculus – few limits…

Allows deductive reasoning only:

- Takes as input the domain-dependent axioms and the set of happened events…
- Provides as output the set of fluents that are true after all the specified events

- What if a new happened event is observed?
- New query is needed: re-computes from scratch the results… computationally very costly !!!!

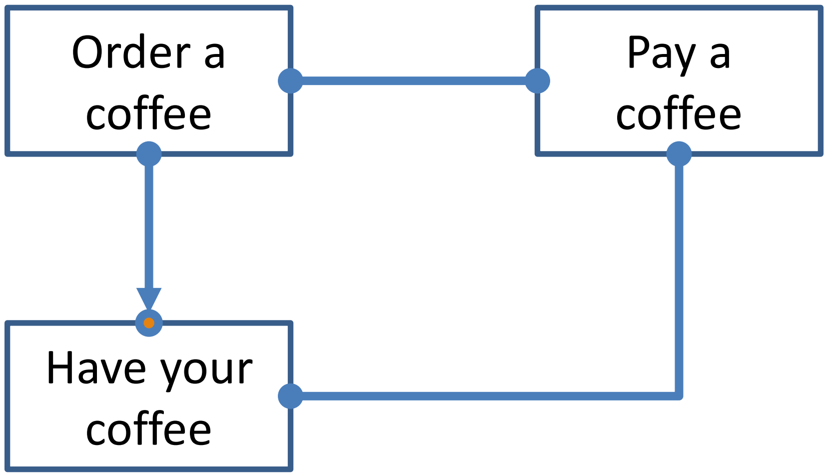
Extensions: Reactive Event Calculus

- Luca Chittaro, Angelo Montanari: **Efficient Temporal Reasoning in the Cached Event Calculus**. Computational Intelligence 12: 359-382 (1996)
- Federico Chesani, Paola Mello, Marco Montali, Paolo Torroni: **A Logic-Based, Reactive Calculus of Events**. Fundam. Inform. 105(1-2): 135-161 (2010)

- Overcome the deductive nature of the original formulation given by Sergot & Kowalski
- New happened events can be added dynamically, i.e. the result is **updated** (and not re-computed from scratch)
- Allows events in a wrong order
- More efficient
- Can be implemented in backward reasoning as well as in forward reasoning

Event Calculus – Application: Monitoring

- If we want the overall system exhibit some robustness …
- … we need to continuously monitor it at run-time
  - Detect unwanted situations asap
  - React accordingly
- Example: Having a coffee at the bar, Declare constraints:



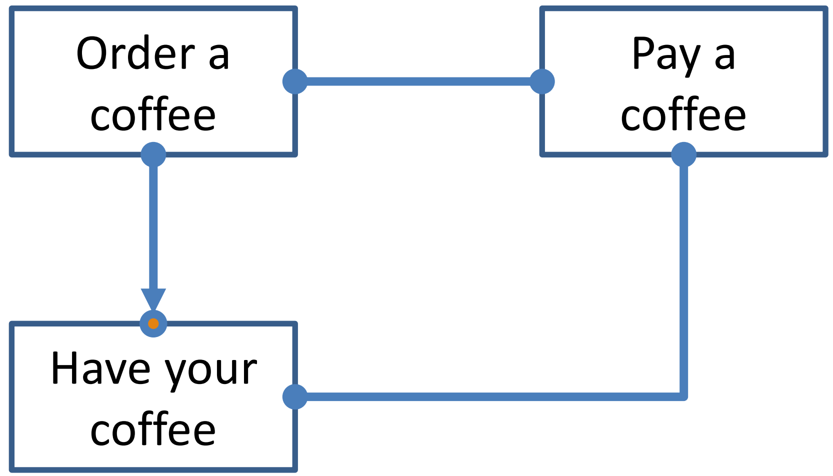
Three events:

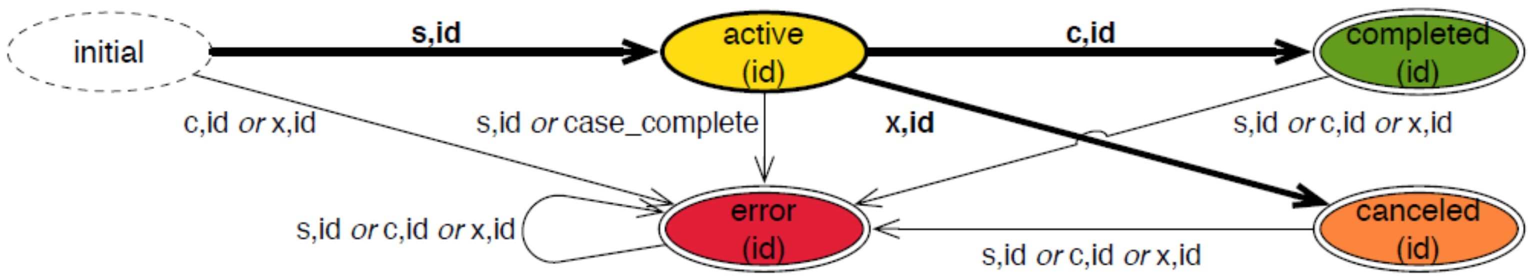
- order
- pay
- have the coffee

Which fluents?

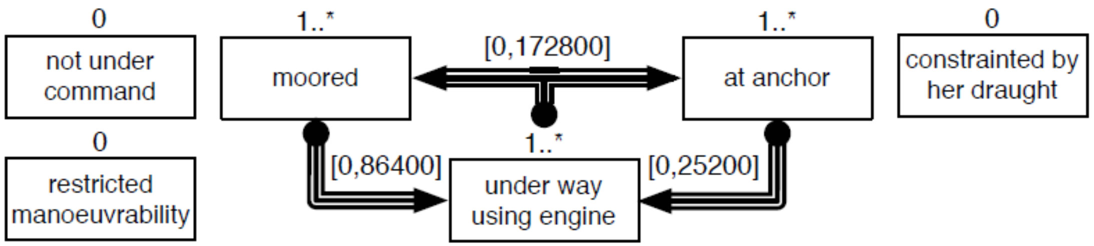
Event Calculus – Application: Monitoring

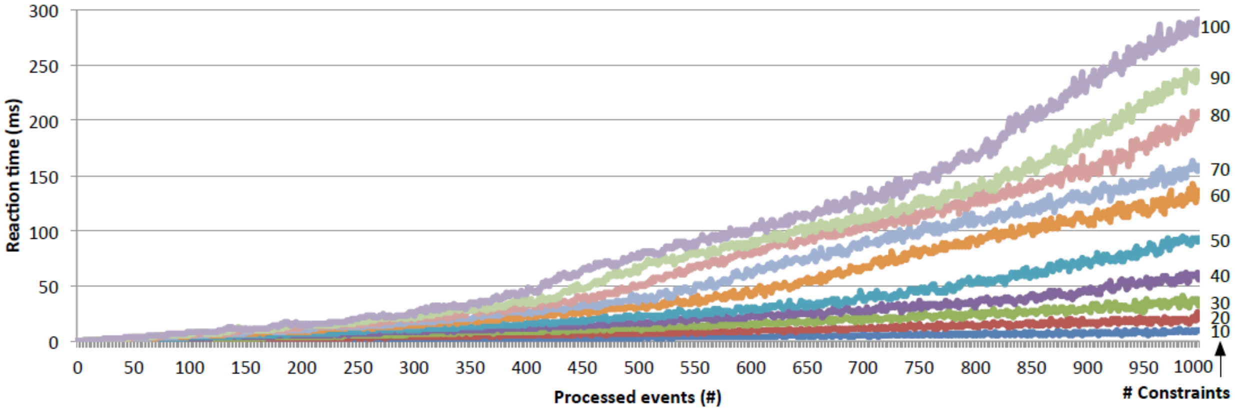
- Represent the status of each constraint through EC ...
- Fluents capture the status of each constraint following a simple finite-state machine
- Possibly, meta-events representing the change of state
- Meta-events connects fluents of different constraints

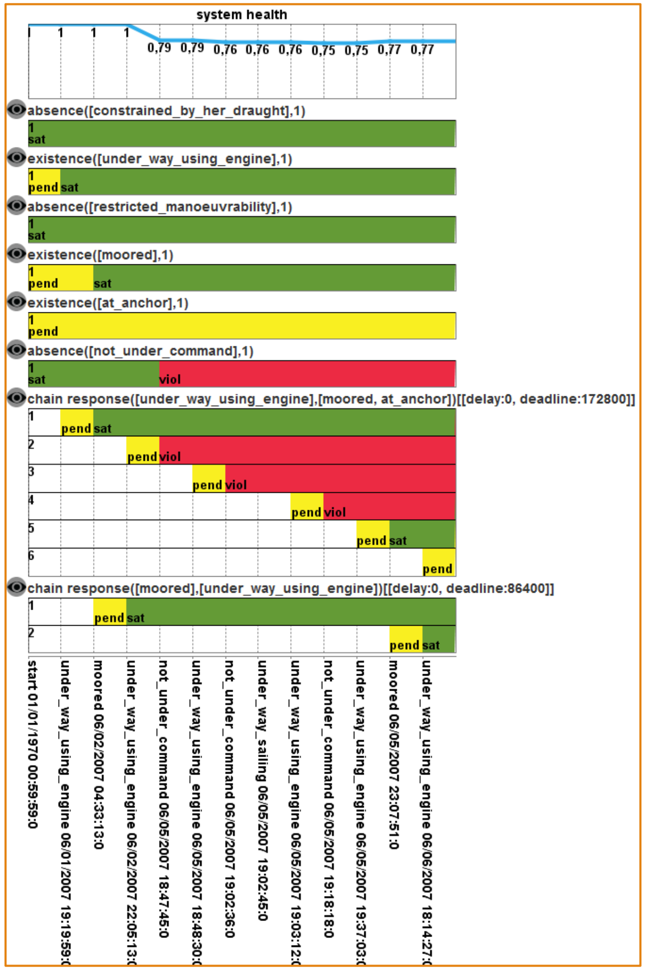




Event Calculus – Application: Monitoring







Allen's Logic for reasoning over temporal aspects

Events…

EC is based on the notion of events or, better saying, on the happening of things

- Are these events instantaneous?
- Or do they have a duration?

Before, how do we measure the passing of time?

- every time scale is an arbitrary reference system
- we need an origin
  E.g.: 1 January 1970
- we can choose a measurement unit
  E.g.: seconds
- points in time are measured w.r.t. distance from the origin

Events…

EC is based on the notion of events or, better saying, on the happening of things

- Are these events instantaneous?
- Or do they have a duration?

The question is not trivial: there are pro&cons for both the choices. **However, there is a consensus that duration-based representation of events is richer.**

Notice:

- a durative event can be represented in terms of start and end time points
- an instantaneous event has duration zero

Allen's logic of intervals

In 1984 (1983?) James Allen proposed to reason about intervals, rather than point in times.

James F. Allen. 1983. Maintaining knowledge about temporal intervals. Commun. ACM 26, 11 (Nov. 1983), 832–843. https://doi.org/10.1145/182.358434

In Allen's view, intervals are more natural categories for humans.

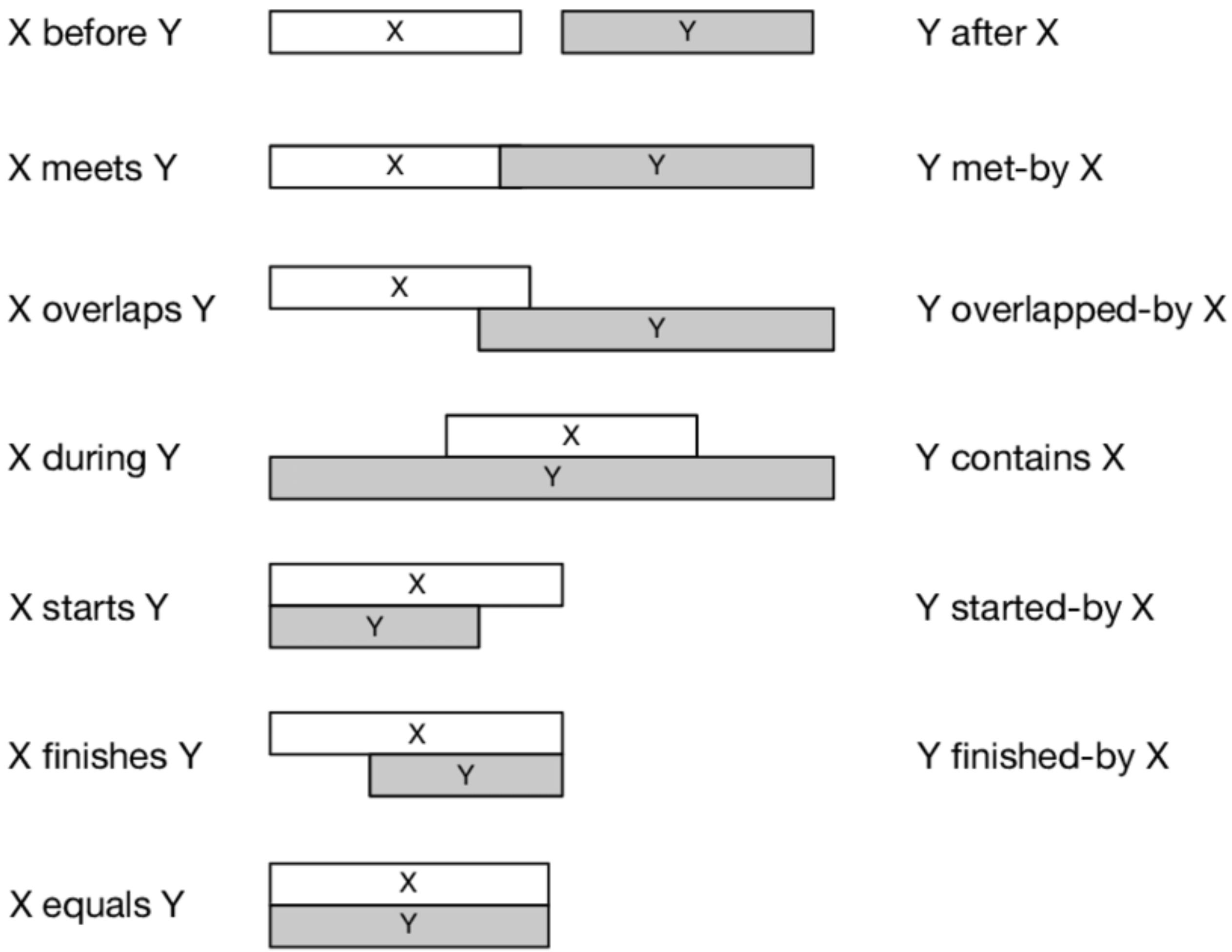
- an *interval i* begins at a certain time point: we introduce the function Begin(i) that return that timepoint
- an *interval i* ends at a certain time point: Ends(i)

Allen's Logic

Allen proposed 13 temporal operators:

- Meet(i,j) ⇔ End(i) = Begin(j)
- Before(i, j) ⇔ End(i) < Begin(j)
- After(j, i) ⇔ Before(i, j)
- During (i, j) ⇔ Begin(j) < Begin(i) < End(i) < End(j)
- Overlap(i, j) ⇔Begin(i) < Begin(j) < End(i) < End(j)
- Starts(i, j) ⇔ Begin(i) = Begin(j)
- Finishes(i, j) ⇔ End(i) = End(j)
- Equals(i, j) ⇔ Begin(i) = Begin(j) AND End(i) = End(j)

Allen's Logic operators – graphically



- Taken from: Colonius, Immo. (2015). Qualitative Process Analysis : Theoretical Requirements and Practical Implementation in Naval Domain.

Allen's Logic operators – graphically

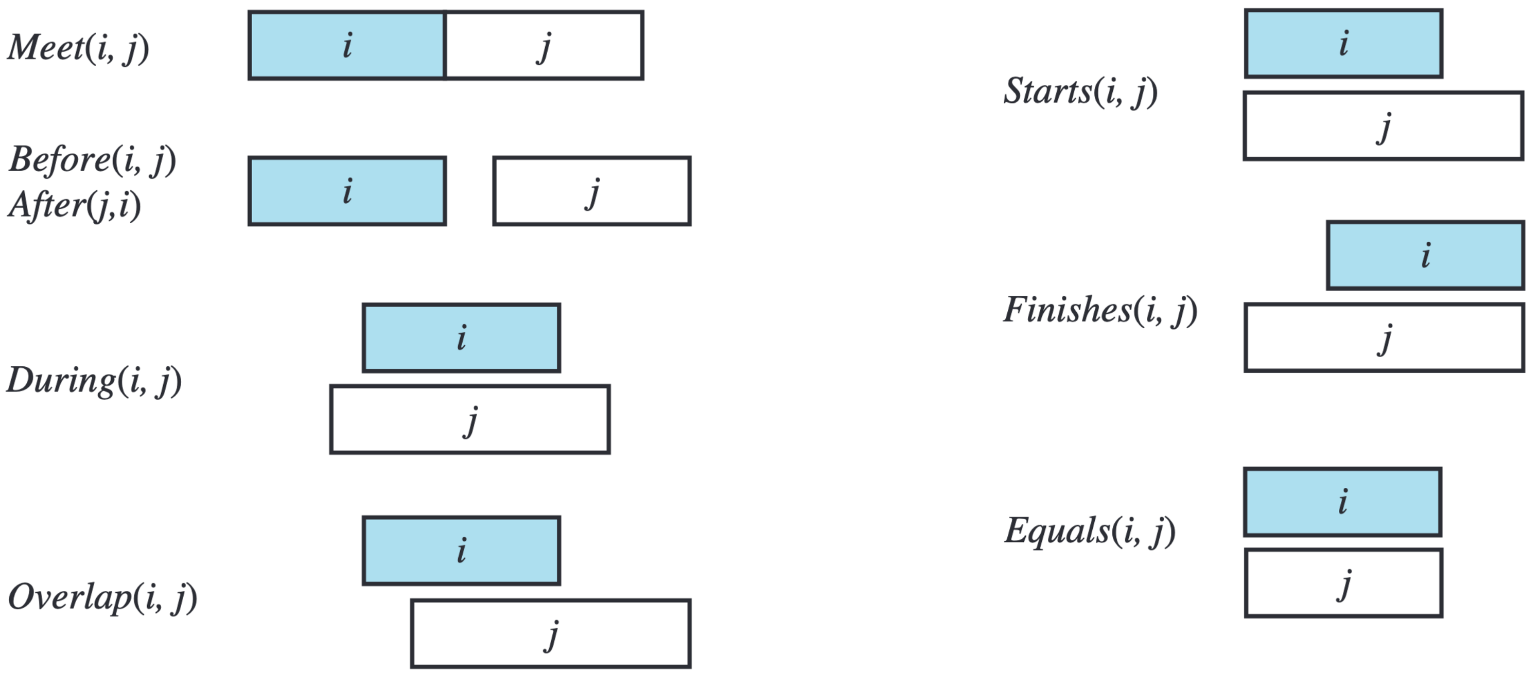


Figure 10.2 Predicates on time intervals.

- Taken from AIMA

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

federico.chesani@unibo.it