Lecture 5: Descriptions, continued

Kenneth M. Anderson
Object-Oriented Analysis and Design
CSCI 6448 - Spring Semester, 2001

Goals for this Lecture

- Discuss Events and Intervals
- Discuss generic characteristics of descriptions
  - In particular
    - Scope and Span

Dynamic Domains

- Dynamic Domains have state
  - This state changes over time
  - To model this change, we introduce
    - events
    - intervals
  - To model states, without worrying about state changes, we can use
    - points in time

Points in Time

- p is a point in time ≈ TimePoint(p)
- The point p precedes the point q in time ≈ Precedes(p,q)

- This way of treating time lets you talk about states, about what is true in the world at a particular point in time
  - but its not particularly helpful in talking about state changes

A TimePoint Example

- A Traffic Intersection
  - The North-South light is red at time p $\approx$ NSRed(p)
  - The North-South light is green at time p $\approx$ NSGreen(p)
  - The East-West light is red at time p $\approx$ EWRed(p)
  - The East-West light is green at time p $\approx$ EWGreen(p)
- An associated description
  - $\forall$ p • (TimePoint(p) $\rightarrow$((NSRed(p) $\vee$EWRed(p)) $\wedge$ (NSRed(p) $\rightarrow$ $\exists$ q • (Precedes(p,q) $\wedge$ NSGreen(q))) $\wedge$ (EWRed(p) $\rightarrow$ $\exists$ q • (Precedes(p,q) $\wedge$ EWGreen(q)))))

Events and Intervals

- If we want to model changes of state, we need to think about intervals of time that are split by events
  - events are atomic (they cannot be interrupted) and instantaneous (time does not progress during an event)
  - intervals occur between events
    - state does not change during an interval

Events and Intervals, Continued

- Designations
  - e is an atomic instantaneous event $\approx$ Event(e)
  - The event e occurs before the event f $\approx$ Earlier(e,f)
  - v is an interval of time in which no event occurs $\approx$ Intvl(v)
  - The event e begins the interval v $\approx$ Begins(e,v)
  - The event e ends the interval v $\approx$ Ends(e,v)
- Definitions
  - InitIntvl(v) ♦ Intvl(v) $\wedge \neg$ $\exists$ e • Begins(e, v)
  - FinalIntvl(v) ♦ Intvl(v) $\wedge \neg$ $\exists$ e • Ends(e, v)
- Note: the definitions make InitIntvl and FinalIntvl available without implying that they are true of any individual interval

Discussion

- In this model, events cannot occur at the same time
  - Earlier(a, b) means that a occurs before b
  - The restriction is that an event represents a transition of the world from one state to another without passing through any intermediate state
  - This means that any action that has an internal time structure must be regarded as two or more events
  - In addition, abstraction plays a key role
    - what the appropriate “event” for a chess game?

Bank Example

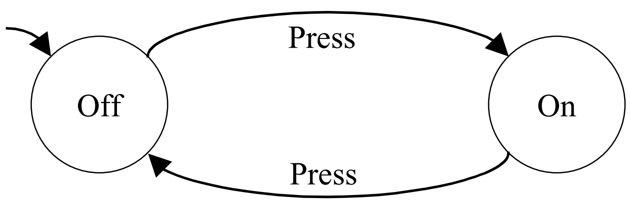
- In event e, account a is debited cash amount m ≈ Debit(e,a,m)
- In event e, account a is credited cash amount m ≈ Credit(e,a,m)
- How do you treat a transfer?
  - As a single transaction?
  - As two events?
  - What about a transfer in which the debit account and the credit account are the same?
    - Does the balance of the account dip?

Modeling State Changes

- In an event-interval view of the world, nothing changes without an event
- If a fact does not reference an interval, then it is a phenomena that does not change
  - Lecturer t teaches course c ≈ Teaches(t,c)
- In an event-interval system, this would imply that teacher t has taught course c eternally
- To give our teachers a break, try
  - Lecturer t teaches course c in interval v ≈ Teaches(t,c,v)

An Event-Interval Example

- A light bulb
  - In interval v, the light is on ≈ On(v)
  - In interval v, the light is off ≈ Off(v)
  - In event e, the button is pressed ≈ Press(e)



Associated Descriptions

- Switch must be on or off in any particular interval, its initial state is off
  - $\forall$ v • ((On(v) → ¬Off(v)) ∧ (Intvl(v) → (On(v) ∨ Off(v))) ∧ (InitIntvl(v) → Off(v)))
- The light changes state via Press events
  - $\forall$ v,w,e • ((Ends(e,v) ∧ Begins(e,w)) → (((On(v) ∧ Off(w)) ∨ (Off(v) ∧ On(w)))) ↔ Press(e))

Why do we need descriptions?

- Why do we need more than one type?
  - You will not be able to express everything about a software problem using one type of description
  - The complexity of software problems do not allow you to think about the whole problem at once
    - So, identify distinct aspects and describe them separately; this is known as **separation of concerns**
    - there are many ways of establishing this separation
      - Can you identify some examples?
      - Note: modularizing your requirements or specifications is at least as important as modularizing your programs

Scope

- A useful description has a carefully defined scope
  - What domain are you describing?
  - What phenomena are you describing?
  - Example: maps
    - Map of the earth or map of the moon?
    - Rainfall map or Population map?
      - For the former, you might create a designation like "The average annual rainfall in area a is r ≈ rainavg(a,r)"

More on Scope

Descriptions are connected to domains by designation sets; the designation set provides the vocabulary the description can use in describing the domain

The designation set, in fact, determines the scope of the description, since you cannot talk about non-designated phenomena

An example

- x is a company employee in time interval y ≈ Comp(x,y)
- x is a time interval in the life of the company ≈ Intl(x)
- x is the manager of y in interval z ≈ Superior(x, y, z)

- A description can talk about the relationships among these phenomena, *but no others!*
- Comp(x,y) is not true unless Intl(y) is true; and
- Superior(x,y,z) is not true unless both Comp(x,z) and Comp(y,z) are true; and
- Superior(x1,y,z) and Superior(x2,y,z) are not both true unless x1 and x2 are the same individual; and
- In any interval z, there is exactly one individual y for which Super(x,y,z) is false for every x

More on Scope

- One way to think of scope is by analogy with parameterized program procedures
  - a procedure has formal parameters, which are replaced by actual parameter values when the procedure is called
  - a description also has formal parameters: they are the terms it uses to refer to the phenomena whose relationships it describes
  - Description D1 (Comp(x,y), Intl(x), Superior(x,y,z))

Implication

- The same descriptions can be applied to different phenomena
- For example, without changing our four descriptions of the company in the previous example slide, we can apply the description to the domain of tennis tournaments
  - We simply change the meaning of the designated phenomena
  - The descriptions remain true; same description, different scope; (see example next slide)

An example revisited

- x is a competitor in the tournament y ≈ Comp(x,y)
- x is an international tennis tournament ≈ Intl(x)
- x beats y in tournament z ≈ Superior(x, y, z)

- The following descriptions still apply!
  - Comp(x,y) is not true unless Intl(y) is true; and
  - Superior(x,y,z) is not true unless both Comp(x,z) and Comp(y,z) are true; and
  - Superior(x1,y,z) and Superior(x2,y,z) are not both true unless x1 and x2 are the same individual; and
  - In any interval z, there is exactly one individual y for which Super(x,y,z) is false for every x

Span

- A useful description also has a span
  - which is the particular subset of the designated phenomena that it includes
  - return to the Map example
    - for a population map, we need to know what part of the world’s surface is depicted; and
    - we need to know whether it is the population in 1990 or in 1800 that is shown

Span Example

- "Thank you for your call. We value it highly. It will be answered in the order in which it was received."
- What's wrong with the proceeding sentence?
  – Hint: "All calls are answered in the order in which they are received."
- The scope selects a domain's individuals; the span selects a subset of those individuals
  – limiting a description's span can help to focus your attention on particular sub-problems

Setting Span

- Scope and Span are closely related;
  – in fact you can express a reduced span by introducing new terms that help to narrow your scope
  – For example
    - x is a citizen of country y ≈ CountryCitizen(x,y)
    - dCitizen ♦ CountryCitizen(x,Denmark)
  – Using dCitizen in a description reduces its span to the citizen's of Denmark (rather than the citizens of the entire world)

Setting Span: Rule of Thumb

- Choose a span of description that allows you to say exactly what you want to say
  – A smaller span would prevent you from saying everything you want
  – A larger span would force you to say too much
    - Saying too much reduces the context independence of a description

Scope and Span Benefits

- The concepts of scope and span can be very useful to a designer
  – When in the process of design, you can ask questions such as
    - Am I making the right kind of separation?
    - Am I using the wrong scope?
    - Am I using the wrong span?
  – Sometimes a problem seems hard, when in fact it is an easy problem surrounded by irrelevant and confusing material

An example

- In a Knock-Out Tennis Tournament, if the number of competitors is a power of 2 —say, 2 to the *n*th power—then there will be n rounds. In each round, half of the players in the round are eliminated, and the other half go forward to the next round, in which there will be only half as many matches. Eventually, there will be a round with only one match, and the winner of that match is the winner of the tournament. If the number of competitors is not a power of 2, then some of them miss the first round, and proceed directly to the second round; from the second round onwards, everything continues as normal.
- The question is: If there are 111 competitors, how many matches are there in the whole tournament?

Questions

- What's the smallest possible scope relating competitors and matches?
- What's the smallest span?
- What's the problem with the proceeding example? Hint: it has to do with its scope and span