

# Semantics of Actions, Agents, and Environments

David Morley

May 1999

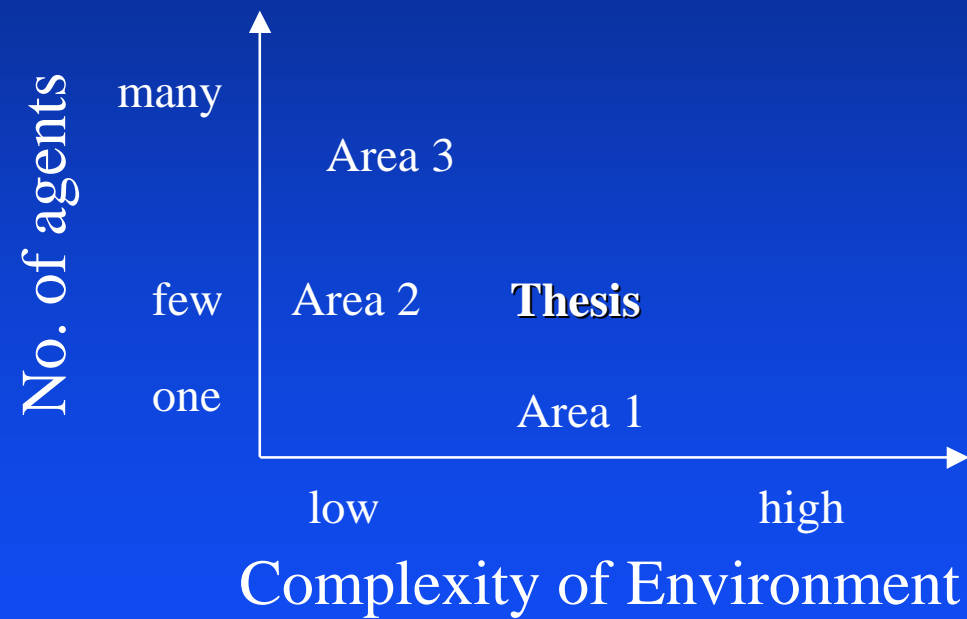
Supervisors:

Liz Sonenberg, Mike Georgeff

# The Problem

- Recent paradigm - agent-oriented systems
- Agent-environment interaction fundamental
- Poor understanding of semantics
  - ◆ agent
  - ◆ environment
  - ◆ interaction

# Agent-Environment Research



# The Contribution

- New formalism for reasoning about action and change
- Application of formalism to semantics of agent-environment systems

# Overview

- Reasoning About Action and Change
- New Event Logic
- Simple BDI Agent Model
- Discussion

# Reasoning About Action & Change

- Model the world and how it changes
- Additional entities
  - ◆ states
  - ◆ fluents
  - ◆ events & actions
- Aim to make predictions/deductions
  - ◆ E.g., Temporal Projection Problem: “Given an initial description of the world, the occurrence of some events, and a notion of causality, what is true after the events have occurred”

# Example: Yale Shooting Problem (Hanks & McDermott 1986)

- Concerns Mary attempting to shoot Fred
- Fluents:
  - ◆  $\text{alive}(F)$  - Fred is alive
  - ◆  $\text{loaded}(G)$  - the gun is loaded
- Events:
  - ◆  $\text{load}(M,G)$  - Mary loads the gun
  - ◆  $\text{wait}(M)$  - Mary waits
  - ◆  $\text{shoot}(M,G,F)$  - Mary aims the gun at Fred and pulls the trigger at point blank range

# Some Issues

## ■ Tractability problems

- ◆ expressing large numbers of facts: non-effects, consequential effects

## ■ Qualification problems

- ◆ ability to reason from incomplete information: many preconditions, possible external events

## ■ Concurrency

- ◆ ability to handle concurrent events
- ◆ desire to provide compositional semantics

# Non-monotonic Approaches

## ■ Proofs:

- ◆ deductions from the absence of information
- ◆ defeasible deductions

## ■ Validity:

- ◆ certain models are preferred over other models

## ■ Examples:

- ◆ circumscription (McCarthy 77)
- ◆ chronological minimisation (Shoham 88)
- ◆ negation as failure (Kowalski & Sergot 86)

# YSP (Hanks & McDermott 86)

## ■ Simple NM Formulation of YSP :

- ◆ introduce abnormality predicate  $Ab(f,e,s)$
- ◆  $Holds(f,s) \ \& \ \sim Ab(f,e,s) \ \rightarrow \ Holds(f, \text{Result}(e,s))$
- ◆  $Holds(\text{loaded}, \text{Result}(\text{load},s))$
- ◆  $Holds(\text{loaded},s) \rightarrow Ab(\text{alive}, \text{shoot}, s)$
- ◆  $Holds(\text{loaded},s) \rightarrow Holds(\text{dead}, \text{Result}(\text{shoot},s))$
- ◆  $Holds(\text{alive},s_0), s_1=\text{Result}(\text{load}, s_0),$   
 $s_2=\text{Result}(\text{wait},s_1), s_3=\text{Result}(\text{shoot},s_2)$
- ◆ “minimise”  $Ab$

# Problem (cont.)

## ■ Model 1

- ◆ s0: alive
- ◆ s1: alive&loaded, Ab(alive,shoot,s1)
- ◆ s2: alive&loaded, Ab(alive, shoot, s2)
- ◆ s3:

## ■ Model 2

- ◆ s0: alive
- ◆ s1: alive&loaded, Ab(alive, shoot, s1) , Ab(loaded, wait, s1)
- ◆ s2: alive
- ◆ s3: alive

# Problems (cont.)

## ■ Led to:

- ◆ a cycle of new logics & new counter-examples
- ◆ lots of analysis of range of applicability

## ■ Non-standard semantics

## ■ Inherently non-compositional

- ◆ cannot combine deductions about sub-systems
- ◆ cannot compose concurrent events

# Monotonic Approaches

- Explicitly state that all *relevant* information is given
  - ◆ deductions come from explicit information, rather than properties of the logic
  - ◆ deductions not defeasible
- Challenge: compactness

# Explanation Closure (Schubert94)

## ■ Main monotonic technique

- ◆ State all possible explanations for change (Explanation Closure axioms)
- ◆ State that no such event occurred
- ◆ Deduce that no change occurred

## ■ Tractability:

- ◆ compact representation ala predicate completion
- ◆ only state what *does* affect a fluent

## ■ More “robust” than N.M. approaches

# Explanation Closure & YSP

## ■ Example effect axiom:

- ◆  $\sim\text{Holds}(\text{loaded}, \text{Result}(e, s)) \leftarrow e = \text{shoot}$
- ◆ shoot loaded become false

## ■ Example Explanation Closure axiom:

- ◆  $\sim\text{Holds}(\text{loaded}, \text{Result}(e, s)) \rightarrow$   
 $e = \text{shoot} \mid \sim\text{Holds}(\text{loaded}, s)$
- ◆ *only* shoot make loaded become false

## ■ We can deduce (from wait $\sim =$ shoot):

- ◆  $\text{Holds}(\text{loaded}, s) \rightarrow \text{Holds}(\text{loaded}, \text{Result}(\text{wait}, s))$

# Explanation Closure Problems

- E. C. axioms too strong
  - ◆ e.g., cannot now introduce unload event without retracting old E.C. axioms
- Difficult to apply to compound events
  - ◆ large number of compound events, some of which do affect the fluents (e.g., shoot||chewGum)

# New Event Logic: Aims

- monotonic
- compositional
- clean semantics
- avoid undesirable consequences

# New Event Logic: Results

- monotonic - yes
- compositional - yes
- clean semantics - yes F.O.P.C.
- avoid undesirable consequences - yes

# New Event Logic: Structure

## ■ Raw event logic

- ◆ new concept of event instance
- ◆ axiomatic description
- ◆ first-order predicate calculus semantics

## ■ Enhanced event logic

- ◆ new syntactic types - event types, actions
- ◆ abbreviations for event instance expressions
- ◆ proof by algebraic manipulation

# What is an Event? Possible view

## ■ Behaviour - a sequence of states

- ◆ E.g. Yale Shooting Problem (a=alive,l=loaded) one behaviour is [(a), (al), (al), ()]

## ■ Event as set of behaviours?

- ◆ E.g. load = {[(a), (al)], [(), (l)]}, etc.
- ◆ Problems when dealing with concurrent events
  - ◆ Behaviours where event performed alone
  - ◆ Behaviours where other events occur as well
  - ◆ Can't derive one from the other

# Introduce Event Instance

- Consider a behaviour where an event occurs
  - ◆ E.g., this talk - “presenting a talk” event occurs
- Not all changes relate to event occurrence
  - ◆ E.g., change in price of BHP shares
- Take event instance to be behaviour + indication of relevant changes

# Example Interpretation

## ■ Annotate sequence of states with changes

### ◆ E.g., load includes

- ◆  $[(\text{O})+1(\text{I})]$  - Fred dead and gun caused to be loaded
- ◆  $[(\text{a})+1(\text{al})]$  - Fred alive and gun caused to be loaded

## ■ Some changes may not be part of event

### ◆ E.g., load also includes

- ◆  $[(\text{a})+1(\text{I})]$  - Fred happens to die during load
- ◆  $[(\text{O})+1(\text{al})]$  - Fred resuscitated during load
- ◆ *Can't* deduce what happens to “a” from a statement “load occurs”

# An event “on its own”

- Can derive behaviour of event on its own:
  - ◆ Given a set of fluents: select instances where all changes in those fluents are effects
  - ◆ E.g., load occurring on its own (with respect to alive and load):  $\{ [() + 1(1)], (a) + 1(al) \}$
  - ◆ Note:  $[(a) + 1(1)]$  and  $[() + 1(al)]$  excluded since change in “a” not caused by instance
- *Can* deduce that “a” stays the same from statement “load occurs on its own”

# Benefits

- Tractability problem
  - ◆ “system” solution presented in thesis
- Qualification problem
  - ◆ State non-effects of events
  - ◆ State events that happen
  - ◆ State that these events occur “alone”
- Monotonic
- Avoids Explanation Closure Axioms
  - ◆ no statements about other events required

# Combining Event Instances

## ■ Can combine effects: “+” operator

- ◆ Only if behaviours identical

- ◆ E.g.,  $[(a)+1(1)] + [(a)-a(1)] = [(a)+1-a(1)]$  - load and die concurrently

## ■ Can concatenate behaviours: “;” operator

- ◆ Only if end & start states match

- ◆ E.g.,  $[(a)+1(a1)] ; [(a1)-a(1)] = [(a)+1(a1)-a(1)]$  - load and then die

# Raw Event Logic

## ■ Types:

- ◆ fluents, (e.g., a, l)
- ◆ fluent values, (e.g., true, false)
- ◆ event instances, (e.g., i1+i2, i3;14)

## ■ Predicates:

- ◆ Relate fluents and event instances
  - ✦ E.g.,  $\text{val}(a, \text{true}, [(a)+l(a)])$  ,  $\text{nab}(a, [(a)+l(1)])$
- ◆ Event types
  - ✦ E.g.,  $\text{load}(i1)$ ,  $\text{wait}(i2)$

## ■ Semantics:

- ◆ first order predicate calculus

# Not Syntactically Convenient

## ■ Verbose:

- ◆ e.g. “`i=i1;i2;i3 & load(i1) & wait(i2) & shoot(i3) & i` occurred”
- ◆ cumbersome dealing with event instances

## ■ Event *types* more natural description:

- ◆ e.g., “instance of `(load;wait;shoot)` occurred”
- ◆ problem - event types are second-order

# Solution

- Event types as abbreviations/macros
  - ◆  $\mu(\text{load}, i) \equiv \text{load}(i)$
  - ◆  $\mu(E1;E2, i) \equiv i=i1;i2 \ \& \ \mu(E1,i1) \ \& \ \mu(E2,i2)$
  - ◆  $\mu([\text{alive}], i) \equiv \text{val}(\text{alive}, \text{true}, i)$
- Allows concise description of scenarios
  - ◆  $\text{YSP} \subseteq \text{load};\text{wait};\text{shoot}$
  - ◆  $\text{YSP} \subseteq [\text{alive}]^\top$
  - ◆  $\therefore \text{YSP} \subseteq \Sigma;[\sim\text{alive}]$

# Fluent Events

## ■ Event types describing fluents

- ◆  $f$  has value  $v$
- ◆  $f$  changes value from  $v1$  to  $v2$
- ◆ no changes to  $f$  are effects
  - ◆ used to state non-effects of events
- ◆ all changes to  $f$  are effects
  - ◆ used to state an event “occurs alone”

# Event Operators

- ◆ concatenation : E1 occurs and then E2 occurs
- ◆ intersection : event of type E1 and E2 occurs
- ◆ concurrent composition : E1 occurs and in parallel a different E2 occurs
- ◆ co-occurrence : E1 occurs and E2 occurs
- ◆ union : event of type E1 or E2 occurs
- ◆ negation : event not of type E occurs

# Not First Order?

- Event types not first-order objects

- ◆ no quantification

- Proof by algebraic manipulation

- ◆ algebraic properties derived from instance axioms and abbreviation definitions
- ◆ e.g.,  $E1;(E2;E3) = (E1;E2);E3$

# Actions

- May be successful or unsuccessful
  - ◆ e.g., attempt to fire gun
- Related to event types
  - ◆ event type of successful execution
  - ◆ event type of failed execution
- Described by set of event instances (possible executions) with results attached
  - ◆ e.g.  $\{ [(a) \text{-} l \text{-} a()]:S, [(a), (a)]:F, \dots \}$
  - ◆ instance may be success for one action fail for another action

# Action Operators

- Build complex actions from simple actions
  - ◆ conditional - if/then/else
  - ◆ iteration - while do
- Correspond to programming constructs

# BDI Agents

## ■ Internals

- ◆ Beliefs - information about the world
- ◆ Desires - goals to achieve
- ◆ Plans - means for achieving goals
- ◆ Intentions - plans committed to

## ■ Interactions

- ◆ Beliefs updates by perception of environment
- ◆ Environment modified by actions

# Simplifying Assumptions

- No parameters on fluents
- Primitive actions restricted to belief changes
  - ◆ Execution of plans is a purely mental process
  - ◆ Separate sensor processes update beliefs
  - ◆ Separate effector processors cause environment actions when triggered by beliefs
- Simple plan triggers
  - ◆ new belief - asynchronous
  - ◆ sub-goal - synchronous

# Semantics of Plan Execution

- Defined using new event logic
  - ◆ plans = compound actions
  - ◆ triggering asynchronous plans = causal process
- Independent of environment

# Semantics of Interaction

- Treat environment, sensors/effectors, plan execution as separate processes
- Compositional semantics of event logic defines parallel composition of processes
- No problem dealing with multiple agents

# Implementation

- Thesis contains implementation of simple BDI agent model in Python
  - ◆ adds some features
  - ◆ about 20 pages

# Application of Formalism

- New simple BDI agent model with well-defined semantics
- Capability to reason about agent-environment interactions
- Capability to reason about multiple agents
- Opens up new area of agent-environment research

# Discussion

- Summary
- Future Directions

# New Formalism Features

- Rich language of events
- Compositional semantics
- Well-understood semantics (FOPC)
- Avoids introducing E. C. axioms
- Handles
  - ◆ concurrency & other compound events
  - ◆ counterfactuals
  - ◆ causality
  - ◆ action failure

# Future Directions

## ■ Event logic:

- ◆ handle continuous change
- ◆ handle probabilities rather than possibilities

## ■ Simple BDI agent model

- ◆ parameterised fluents
- ◆ maintenance conditions
- ◆ meta-level reasoning