Lecture W2: The LSD Notation for Agent Specification

W4.0 Background and Motivation

History of LSD and ADM

LSD notation

- first developed by Beynon and Norris of BTRL in 1986
- original influence over design of LSD notation was SDL
- elaboration of the design due to Mike Slade: MSc thesis 1989
- aim of Slade's work animation from LSD specifications
- this led to the design and implementation of the ADM:
  input from Edward Yung re abstract model for EDEN
- graphical animation for ADM by Simon Yung via ADM+EDEN

Characteristics of LSD specification and ADM simulation

- *LSD specification is oriented towards agent-interaction*

  cf user-interface designer not abstract formal specification

  Aim to represent
  - what an agent is responsive to ("observes") in a system
  - how it can affect the state of the system by actions

  cf human factors in system design

  - what does the user need to know?
  - what must the user be able to perceive?
  - what is effect of imperfect knowledge or loss of capability?

  cf engineering design

  - explain system behaviour in terms of component behaviour
    as verified by experiment
  - determine how reliable the communication and
    how fast the response of agents has to be

- *modelling and simulation associated with LSD specification*
  rooted in physical experiment and observation

  ie complex specifications for systems can only be constructed
  by building and experimenting on components

  cf mimicing the global behaviour of a system without explicitly modelling the relationship between its component parts: cf
  formal specification, statistical simulation tools, naive physics

- *analysing agent characteristics in an LSD specification*
  does not *lead directly to an executable model.*
  an operational interpretation ***requires more assumptions***

  - LSD has no formal operational semantics in the CS sense

  - LSD is concerned primarily with requirements analysis:
    identification of system structure
    prior to circumscription of its actual / intended behaviour

. • analysing agent interaction raises significant issues.

W4.1. Form of the LSD specification of an Agent

The LSD specification of an agent comprises 4 kinds of variable:

oracle - a variable to which it responds

state - a variable that it owns

handle - a variable conditionally under its control

derivate - an indivisibly coupled stimulus-response relation

+

protocol = list of privileges of the form

enabling condition → sequence of actions

LSD simulation is from a set of instances of specified agents

each action either

changes a parameter

or invokes / deletes an agent instance

```
agent sm() {                                    // The station master:
state       (time) tarrive = |Time|,            // registers time of arrival
            (bool) can_move = false,            // determines whether driver can start engine
            (bool) whistle = false,             // controls the whistle
            (bool) whistled = false;            // remembers whether he has blown the whistle
            (bool) sm_flag = false,             // controls the flag
            (bool) sm_raised_flag = false       // remembers whether he has raised the flag
oracle      (time) Limit, Time,                 // knows the time to elapse before departure due
            (bool) guard_raised_flag,           // knows whether the guard has raised his flag
            (bool) driver_ready,                // knows the driver is ready
            (bool) around[d] (d = 1 .. number_of_doors),
                        // knows whether there's anybody around doorway
            (bool) door_open[d] (d = 1 .. number_of_doors)      // the doors status
handle      (bool) can_move, whistle, whistled, sm_flag, sm_raised_flag,
            (bool) door_open[d] (d = 1 .. number_of_doors) // partially controls the doors
derivate    (bool) ready = ⋀ (¬door_open[d]) | d = 1 .. number_of_doors),
                        // monitors whether all doors are shut
            (bool) timeout = (Time – tarrive) > Limit; // monitors whether departure due
protocol
            door_open[d] ^ ¬around[d] –> door_open[d] = false, (d = 1 .. number_of_doors)
            ready ^ timeout ^ ¬whistled
                        –> whistle = true; whistled = true; guard(); whistle = false,
            ready ^ whistled ^ ¬sm_raised_flag –> sm_flag = true; sm_raised_flag = true,
            sm_flag ^ guard_raised_flag –> sm_flag = false,
            ready ^ guard_raised_flag ^ driver_ready ^ engaged ^ ¬can_move
                                                    –> can_move = true
}
```

Terminology: illustrated by the VCCS LSD specification

"variable" neutral term for oracles, states, handles and derivates:

same identifier may appear in many agent specifications:

all **observations** by *different* agents of the *same* thing

e.g. measSpeed **state** for speed_transducer

oracle for throttle_manager

oracle for the driver

cruiseStts **state** for cruise_cutout

oracle for the driver

oracle for the throttle_manager.

oracle, state, handle, derivate characterisation

<-> status of the observation with respect to the agent

v-observation = observation designated by identifier v

In simulation – in general:

identifier instances <--> different variables

system designer identifies observations in LSD specification

<--> what would record and subject to experiment to relate

(pre-act-of-faith) behaviour of the system to

(post-act-of-faith) reliable activity of the components

Classification of variables associated with an agent

states

"variables *bound* to the agent"
= observations associated with an agent that would disappear were the agent instance to disappear

E.g. accel, windF, actSpeed, ... states of vehicle agent

A state variable v is the primary source of all v-observations:
at any time, all v-observations correspond to ≤1 state variable.

state variable v = "authentic value" of v-observations

- agent may not know authentic value of its state variables
  ie variable may be an oracle and a state to the same agent
  eg "Time as recorded by my watch", or "my bank balance".

Observer and actor agents tend to have few state variables
E.g. no state variables associated with driver agent in VCCS

consts

Constants =state variable, value not subject to change

e.g. the physical parameters of the vehicle: mass, windK, rollK.

In the VCCS as implemented, could vary some "constants"

E.g. to take account of loading the vehicle

to relate the wind resistance to the vehicle profile

oracles

In anthropomorphic terms:

an **oracle** = value as perceived by an agent.

NB no guarantee that its value is accurate.

an oracle ≠ an input

e.g. actSpeed is **state** in vehicle though used by a derivate

cf in the privilege

!door_locked ∧ !door_open → door_open = **true**

door is locked not referenced as an oracle to the door use

Whether the door can be opened or not

- independent of perception of whether it is or isn't locked
- the effect of locking the door isn't subject to delay.

handles

handle = parameter agent can manipulate

subject to certain enabling conditions being met

a handle of an agent is not necessarily bound to the agent

E.g. driver agent can change

state variable engineStts of engine agent.

cf an object-oriented modelling paradigm:

where an object invokes a method to change its state

v-handle and v-oracle co-exist=> agent communication

NB need assumptions about communication for animation

e.g, the VCCS specification does not specify how

vehicle speed, as measured by the speed_transducer

is communicated to the throttle_manager

In animation:

- might assume idealised communication
- might refine the specification of the communication channel

protocols

An agent's privileges to act make up its protocol

agent acts subject to enabling conditions

- to redefine handles
- to invoke and delete agent instances

privilege = sequence of actions, conditionally executable

e.g.if engine switched off, the driver can switch on, and v.v.

No fixed set of rules to interpret agent protocols operationally

Considerations include:

- privileges ≠ obligations to act

e.g. in driver protocol; when the cruise controller switched on:

driver can switch it off

request it to maintain the vehicle either

at the specified cruise speed

or at the current measured speed

reset the specified cruise speed.

- privileges can be actions to be executed once enabled

e.g. cruise controller cuts out once the driver touches the brake

- animation to reflect external significance of agent actions

e.g. driver doesn't switch cruise controller on and off frantically

- improbable activity for the driver
- subject to an engineering constraint

W2.2. Principles behind LSD specification for agents

What the LSD specification addresses:

- What are the agents in the system?
- What are the key observations that explain system behaviour?
- How are the observations associated with agents?
- How does an agent register state changes in its environment?
- IWhen does an agent have privileges to change state?
- What observations can an agent change?

The agent as Experimentor

handle	parameter that can be changed
oracle	parameter that can be observed
derivate	perceived dependency between observables

Hooke's Law

change *load* (handle) and observe *extension* (oracle)

States and protocols have a more controversial role

e.g.	protocol	may express a sequence of steps in experiment
	state	may represent characteristic of the experimentor

? Scientific experiment not concerned with subjective observation

Many distinctions to be made between kinds of experiment

Use experimentation for developing conviction, skills, instruments

e.g. timeless (Hooke's Law) vs time-based
cf analogue variables, clocks and events

More about behaviour

Behaviour is a concept that presumes an objective observer

Engineer postulates simultaneous values for all observables
but can't in general simultaneously observe

LSD **doesn't define an unambiguous behaviour** for a system.

Issues for animation, even when presuming an objective stance

- the interpretation of oracles
- the ways in which agents exercise privileges to act
- the nature of the observed stimulus-response relationship
- the speed with which steps in the protocol are executed

e.g. Stationmaster illustrates
derivate as *an indivisible relationship as perceived by an agent*
BUT
derivate as an indivisible relationship as perceived by an agent
≠ definition in animation from an LSD specification

definition in animation ↔ objective FD and synchronisation

Illustrative example

Spreadsheet user updates though cells yet to be made consistent

if o1 is displayed value of cell1 and o2 is displayed value of cell2
then the relation o2 = f(o1) is guaranteed only when up-to-date

o2 is oracle to user to be observed after awaiting update
cf protocol for an experimental observation

Hence spreadsheet user perceives definitive relationships
that aren't universally valid in the view of the objective observer

oracle assumes subtle conventions re communication/observation

User continues to enter data whilst the spreadsheet is still updating on premise that no other agent can interrupt the updating process

Indivisible may not mean *synchronised in time* but *not interruptable*

W2.3. Illustrative Examples

VCCS

Railway Station Animation

Digital Watch

Electronic Catflap
Telephone