Wumpus World Intro

Erik Pitzer

Artificial Intelligence – SS 2021

Java Expert System Shell (1/3)

- declarative programming
- specify rules `defrule`
  - precondition $\Rightarrow$ action
  - rules fire automatically

```
(defrule found-exit
  "If the hunter has gold and finds an exit, she leaves."
  (task act)
  (hunter (agent ?agent) (x ?x) (y ?y) (gold ~0))
  (exit (x ?x)(y ?y))
  =>
  (printout t ?agent " leaves the caves." crlf)
  (halt))
```

Java Expert System Shell (2/3)

- has support for functions
- has support for data structures

```
(deftemplate pit
  (slot x (type INTEGER))
  (slot y (type INTEGER)))
```

Java Expert System Shell (3/3)

- has support for queries

```
(defquery get-adjacent-wumpus-free
    "find all wumpus-free adjacent fields"
    (declare (variables ?x ?y))
    (adj ?x ?y ?x' ?y') ;; cave is adjacent
    (cave (x ?x') (y ?y') ;; hunter knows about this cave
          (has-wumpus FALSE)))
```

- can even call Java

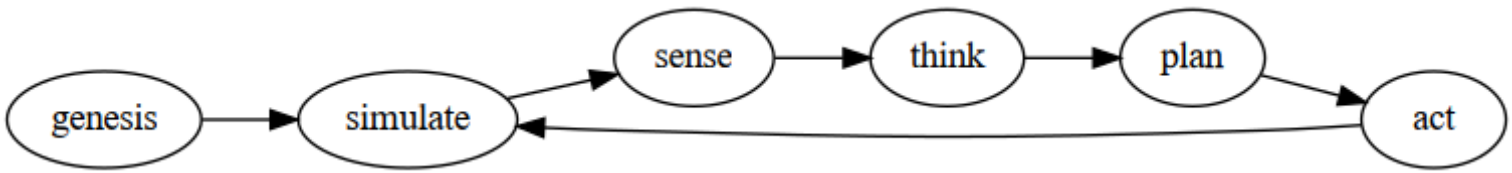
```
(bind ?ht (new java.util.Hashtable)) ;; =>
  ↪  <Java-Object:java.util.Hashtable>
?ht ;; => <Java-Object:java.util.Hashtable>
(?ht put 1 "one")
(?ht get 1) ;; => "one"
```

Problem Description

- Hunter explores a cave
- wants to find gold
- wants to avoid pits
- wants to avoid the Wumpus
- large parts of implementation provided
  - logical reasoning of Hunter is missing

Simulation Engine

- simulation of progress and AI at once
- process different stages



Stage 0: Genesis

- build the world
- data structures
  - `hunter(agent, x, y, gold, alive)`
  - `desire(agent, strength, action, x, y)`
  - `goal(agent, action, x, y)`
  - `nocave(x, y)`
  - `wumpus(x, y, alive)`
  - `pit(x, y)`
  - `gold(x, y, amount)`
  - `exit(x, y)`

Caves

```
(deftemplate cave
  "Cave objects store the hunter's model of the world"
  (slot x (type INTEGER))                ; coordinates of cave
  (slot y (type INTEGER))                ;
  (slot fromx (default -1))              ; coordinates of cave from
    ↪ which
  (slot fromy (default -1))              ;   we first entered the
    ↪ cave.
  (slot visited (default FALSE))         ; Has the hunter been
    ↪ here?
  (slot stench (default UNKNOWN))        ; Does the cave smell?
  (slot breeze (default UNKNOWN))        ; Is it breezy?
  (slot glitter (default UNKNOWN))       ; Is there a glitter in
    ↪ it?
  (slot has-wumpus (default UNKNOWN));Is there a wumpus here?
  (slot has-pit (default UNKNOWN))       ; Is there a pit here?
  (slot has-gold (default UNKNOWN))      ; Is their gold here?
  (slot has-exit (default UNKNOWN))
  (slot safe (default UNKNOWN)))         ; Is the cave safe?
```

- neighbors in the *real* world can be queried using `adj` e.g. for simulation

Stage 1: Simulate

- simulate effects of previous actions or genesis
  - meet the Wumpus

```
(defrule meet-the-wumpus
  "If a hunter and wumpus are in the same cave..."
  (task simulate)
  ?hunter <- (hunter (x ?x) (y ?y) (alive TRUE))
  (wumpus (x ?x) (y ?y) (alive TRUE))
  =>
  (printout t
    ↪  "Aaarrrggghhhhhh....munch...munch...munch"
    ↪  crlf)
  (modify ?hunter (alive FALSE))
  (halt))
```

  - fall into pit

Stage 2: Sense

- perceive effects
  - notice nearby caves

```
(defrule notice-other-caves
  "If we've just entered a new cave, we notice the
    ↪  other adjacent caves."
  (task sense)
  (hunter (agent ?agent) (x ?x)(y ?y))
  (adj ?x ?y ?x2 ?y2)
  (not (cave (x ?x2)(y ?y2)))
  =>
  (printout t ?agent " notices (" ?x2 "," ?y2 ")
    ↪  nearby." crlf)
  (assert (cave (x ?x2)(y ?y2))))
```

  - breeze of pits
  - stench of wumpus
  - glitter of gold

Stage 3: think

- think about perceptions
  - infer safety of cave positions by
  - inferring possible positions of pits and the Wumpus
  - create desires for actions

```
(defrule lust-for-gold
  (task think)
  (hunter (agent ?a) (x ?x) (y ?y))
  (cave (x ?x) (y ?y) (has-gold TRUE))
  =>
  (printout t ?a " wants to pick up the gold in (" ?x
    ↪  "," ?y ")." crlf)
  (assert (desire (agent ?a) (strength ?*veryhigh*)
    ↪  (action pickupgold))))
```

Stages 4 and 5: Plan & Act

- plan
  - pick highest desire
- act
  - perform selected action

Eclipse Integration

- only for eclipse *Kepler*
- does not work with newest JDK
- prepackaged Eclipse with
  - suitable JDK in `/jre` subfolder
  - and Jess features
- available from `Exchange` drive and `FileBox`
  - `\\fshome\Exchange\MSE\LVA\KI2\Jess`
  - `\\fshome\Exchange\IM\LVA\KI2\Jess`
  - `\\fshome\Exchange\MCM\LVA\KI2\Jess`