Modal Logic Simulator (mls)

Iain Moncrief

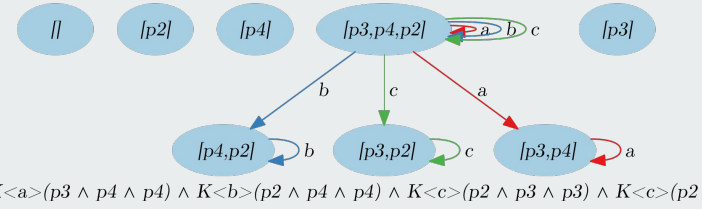
```
-- AST data type for logical formulae
data L agent prim
  = Prim prim
  | Neg (L agent prim)
  | And (L agent prim) (L agent prim)
  | Know agent (L agent prim)
```

```
type Collection a = [a]


type AccessRelation agent state = agent -> Collection (state,state)

type Valuation state prim = state -> prim -> Bool

data KripkeModel agent prim state
  = M { agents        :: Collection agent
      , prims         :: Collection prim
      , states        :: Collection state
      , accessibility :: AccessRelation agent state -- R agent state
      , valuation     :: Valuation state prim       -- V state prim
      }
```

```
type State prim = [prim]

-- Generate Kripke Model and states, satisfying a formula
genKripke :: L agent prim -> (KripkeModel agent prim (State prim),[State prim])

-- Test if a Kripke Model, in a state, satisfies a formula
models :: KripkeModel agent prim state -> state -> L agent prim -> Bool

-- Given a formula (or set of formulas in CNF), test another formula is entailed.
evaluate :: L agent prim -> L agent prim -> Bool
evaluate p p' = and $ map (flip (models km) p') ss
     where (km,ss) = genKripke p
```

$$K_1(\varphi_1 \wedge \ldots \wedge \varphi_n) \wedge \ldots \wedge K_m(\varphi_1 \wedge \ldots \wedge \varphi_k)$$

genKripke



×

$$K_a(\varphi_1 \wedge \varphi_2)$$

models

evaluate

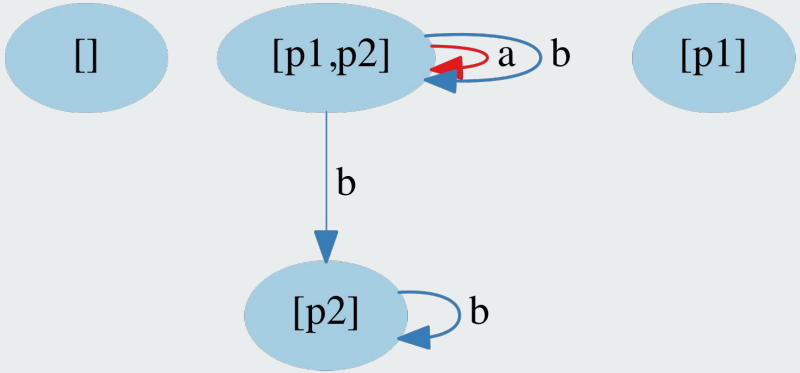
Yes/No

Example

Agent ‘a’ knows (proposition 1 and proposition 2)
Agent ‘b’ knows proposition 2

$$K_a(p_1 \wedge p_2) \wedge K_b(p_2)$$

Expected consequences:
Agent ‘a’ knows proposition 1 $K_a(p_1)$
Agent ‘a’ knows proposition 2 $K_a(p_2)$
*Agent ‘b’ does **not** know (**not** proposition 2)* $\neg K_b(\neg p_2)$



demonstration

(“Any questions?”:[])