Bisimulation

Jan van Eijck

jve@cwi.nl

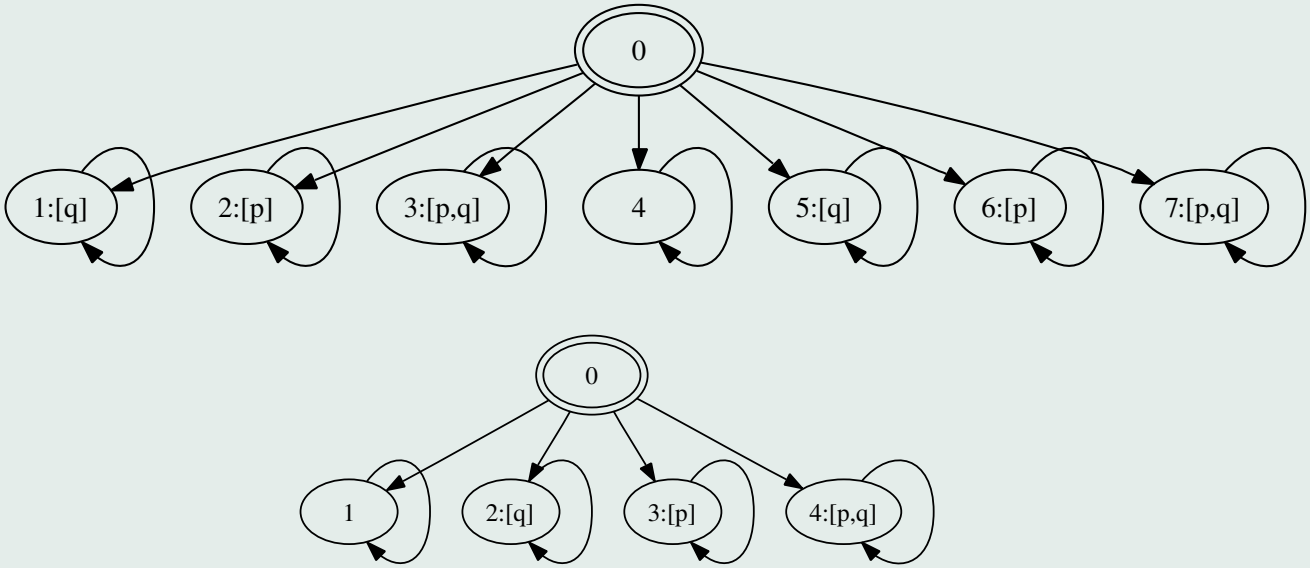
January 9, 2006

Abstract

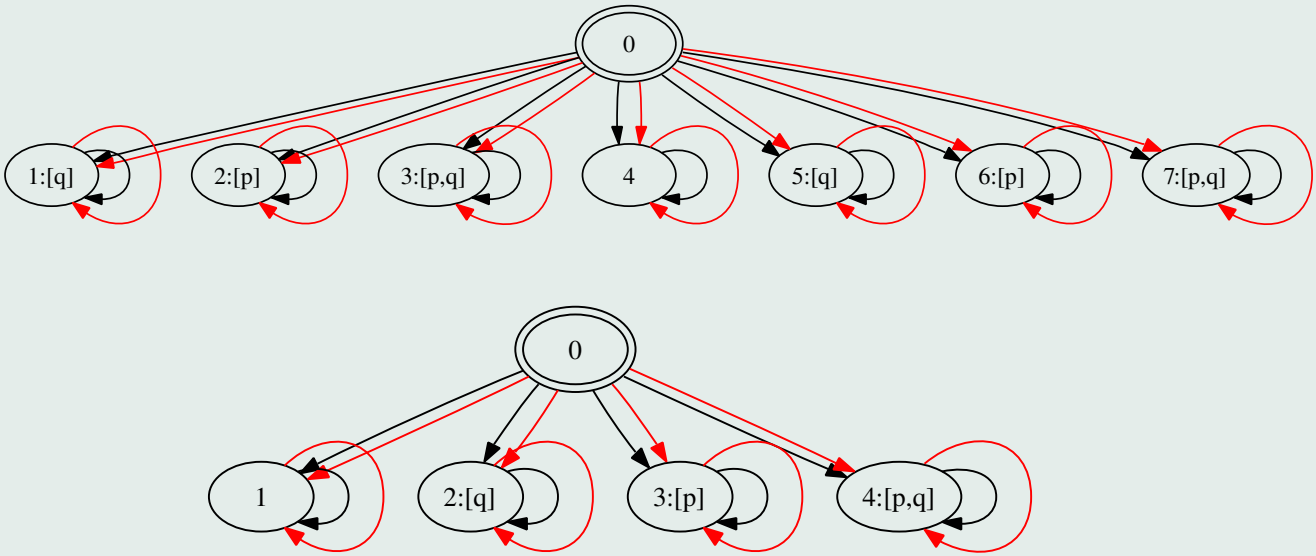
Bisimulation is an important notion in modal logic. Bisimulation is intended to characterize states in Kripke models with 'the same behaviour'.
We will look at examples, then give a definition and prove some theorems.
Finally, we will show how bisimulation can be used to simplify Kripke models by replacing each state in the model by its bisimilarity class, and we will present and implement an algorithm for carrying this out.

When are Epistemic Models 'Equivalent' ?

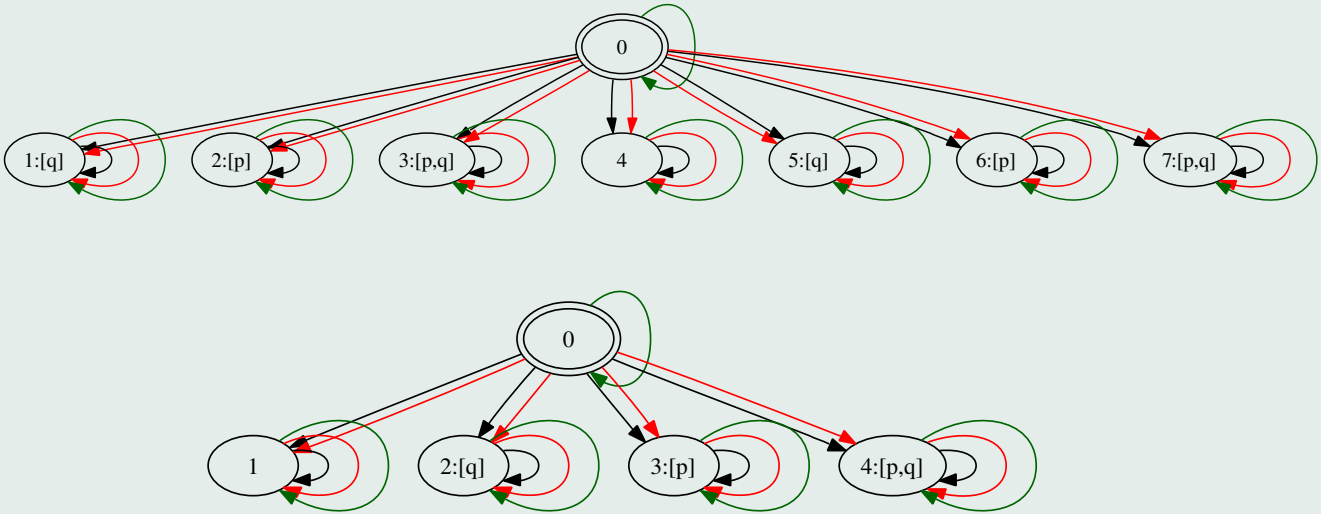
Can you find a formula that is true in the actual world of one of the models, but false in the actual world of the other?



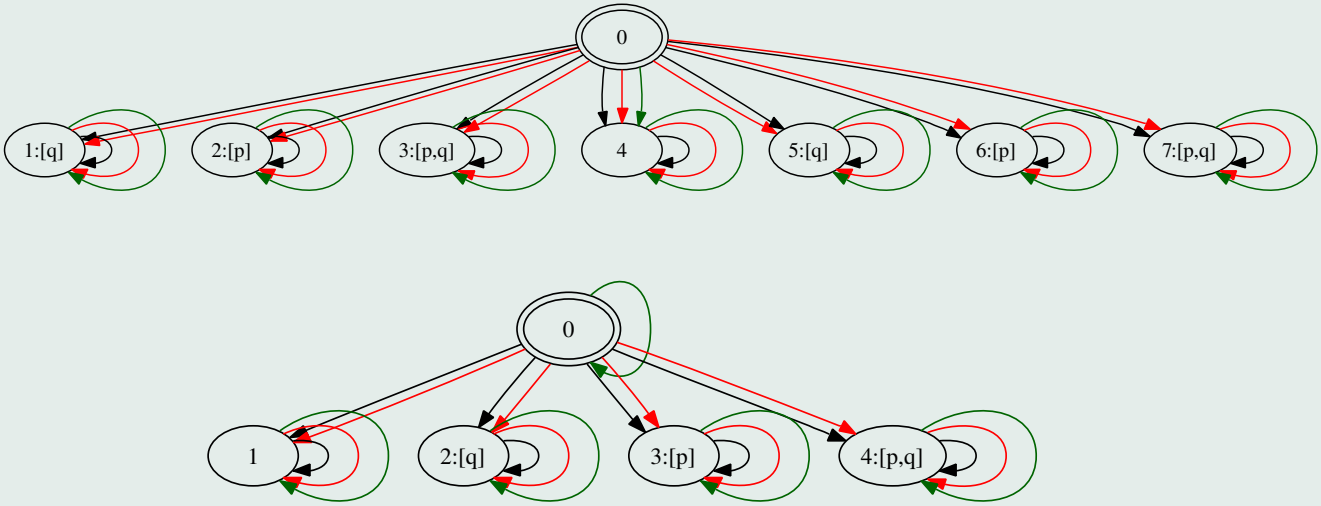
Can you find a formula that is true in the actual world of one of the models, but false in the actual world of the other?



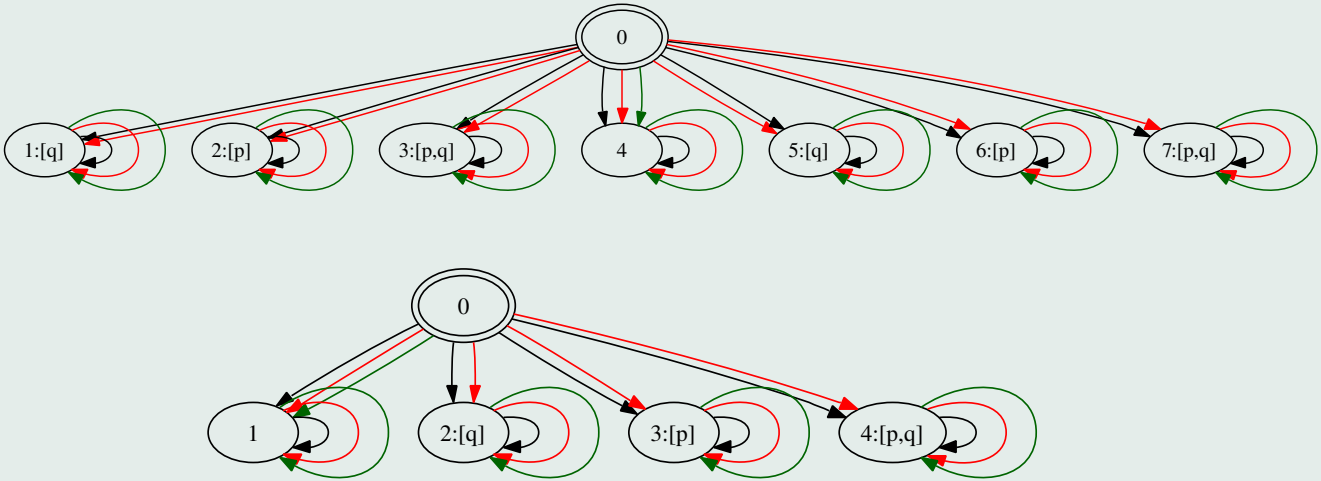
Can you find a formula that is true in the actual world of one of the models, but false in the actual world of the other?



Can you find a formula that is true in the actual world of one of the models, but false in the actual world of the other?

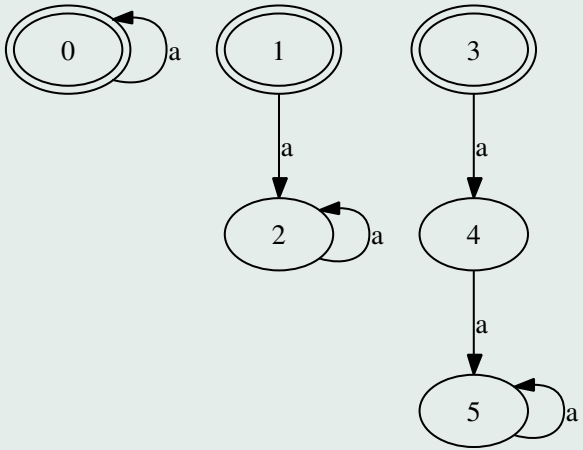


Can you find a formula that is true in the actual world of one of the models, but false in the actual world of the other?



Bisimulation in different contexts

- Modal and epistemic logic (examples above)
- Process theory,



- Set theory (modelling of non-wellfounded sets).

Bisimulation — Definition

The notion of bisimulation is intended to capture state equivalences and process equivalences.

A bisimulation Z between Models $\mathbf{M}$ and $\mathbf{N}$ is a relation on $S_{\mathbf{M}} \times S_{\mathbf{N}}$ such that if sZt then the following hold:

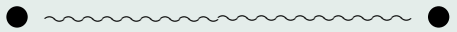
Invariance $V_{\mathbf{M}}(s) = V_{\mathbf{N}}(t)$ (the two states have the same valuation),

Zig if for some $s' \in S_{\mathbf{M}}$ $s \xrightarrow{a} s' \in R_{\mathbf{M}}$ then there is a $t' \in S_{\mathbf{N}}$ with $t \xrightarrow{a} t' \in R_{\mathbf{N}}$ and $s'Zt'$.

Zag same requirement in the other direction.

Use $Z : \mathbf{M}, s \underline{\leftrightarrow} \mathbf{N}, t$ to indicate that Z is a bisimulation that connects s and t. Use $\mathbf{M}, s \underline{\leftrightarrow} \mathbf{N}, t$ to indicate that there is a bisimulation that connects s and t. If the models are clear, use $s \underline{\leftrightarrow} t$. If $s \underline{\leftrightarrow} t$ one says that s and t are bisimilar.

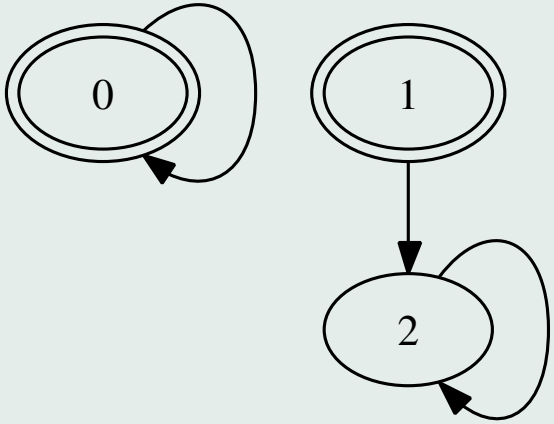
Invariance



Zig

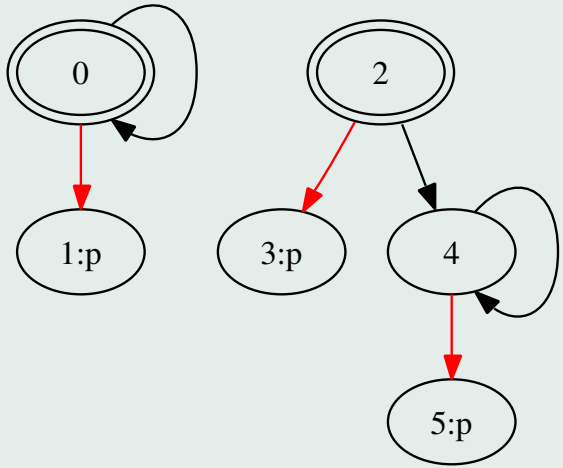
Zag

Bisimulation — Example 1



In the models of the picture, $0 \underline{\leftrightarrow} 1 \underline{\leftrightarrow} 2$.

Bisimulation — Example 2



In the models of the picture, $0 \underline{\leftrightarrow} 2 \underline{\leftrightarrow} 4$ and $1 \underline{\leftrightarrow} 3 \underline{\leftrightarrow} 5$.

Invariance for Bisimulation

A formula φ (of some logical language suitable for talking about Kripke models) is called invariant for bisimulation if the following holds:

If

$$\mathbf{M}, s \underline{\leftrightarrow} \mathbf{N}, t$$

then

$$\mathbf{M}, s \models \varphi \text{ iff } \mathbf{N}, t \models \varphi.$$

Modal Logic

Language for multimodal logic. Assume p ranges over a set of proposition letters and a over a set of relation letters (in the epistemic case, these indicate agents):

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle a \rangle \varphi$$

Abbreviations:

- $\bot$ abbreviates $\neg\top$.
- $\varphi_1 \wedge \varphi_2$ abbreviates $\neg(\neg\varphi_1 \vee \neg\varphi_2)$.
- $\varphi_1 \Rightarrow \varphi_2$ abbreviates $\neg\varphi_1 \vee \varphi_2$.
- $[a]\varphi$ abbreviates $\neg\langle a \rangle\neg\varphi$.

Epistemic logic: read $[a]\varphi$ as $K_a\varphi$. Doxastic logic: read $[a]\varphi$ as $B_a\varphi$.

Modal Logic and Bisimulation

Theorem 1 *Modal formulas are invariant for bisimulation.*

Proof: Let $\mathbf{M}, s \underline{\leftrightarrow} \mathbf{N}, t$. Then there is a bisimulation Z with sZt. We proceed by induction on the structure of φ.

Basic cases: $\varphi = \top$. Then $\mathbf{M}, s \models \varphi$ and $\mathbf{N}, t \models \varphi$, for $\top$ is always true.

$\varphi = p$. From sZt it follows by the invariance condition for bisimulation that s and t have the same valuation. Thus, $\mathbf{M}, s \models p$ iff $\mathbf{N}, t \models p$.

Induction step. $\varphi = \neg\psi$. Then by the induction hypothesis $\mathbf{M}, s \models \psi$ iff $\mathbf{N}, t \models \psi$. It follows that $\mathbf{M}, s \models \neg\psi$ iff $\mathbf{N}, t \models \neg\psi$.

$\varphi = \varphi_1 \vee \varphi_2$: Similar.

Modal case: $\varphi = \langle a \rangle \psi$.

We can assume that the theorem holds for ψ. This is the induction hypothesis.

Assume $\mathbf{M}, s \models \langle a \rangle \psi$. Then there is a s' with $s \xrightarrow{a} s'$ and $\mathbf{M}, s' \models \psi$.

By sZt and the zig condition for bisimulation there is a t' with $t \xrightarrow{a} t'$ and $s'Zt'$. By induction hypothesis $\mathbf{N}, t' \models \psi$.

Therefore $\mathbf{N}, t \models \langle a \rangle \psi$.

Next, assume $\mathbf{N}, t \models \langle a \rangle \psi$. Then there is a t' with $t \xrightarrow{a} t'$ and $\mathbf{N}, t' \models \psi$.

By sZt and the zag condition for bisimulation there is an s' with $s \xrightarrow{a} s'$ and $s'Zt'$. By induction hypothesis $\mathbf{M}, s' \models \psi$.

Therefore $\mathbf{M}, s \models \langle a \rangle \psi$.

Modal Logic and Bisimulation — 2

Theorem 2 (Van Benthem) *If a first order formula φ is invariant for bisimulation, then φ is equivalent to a modal formula.*

Proof: omitted. See, e.g., [1].

What this theorem says is that if first order logic and bisimulation are given, modal logic emerges.

Question about Modal Equivalence

Two states (in the same Kripke model, or in different Kripke models) are modally equivalent if no modal formula can see a difference between the states.

Use $\leftrightsquigarrow$ for modal equivalence.

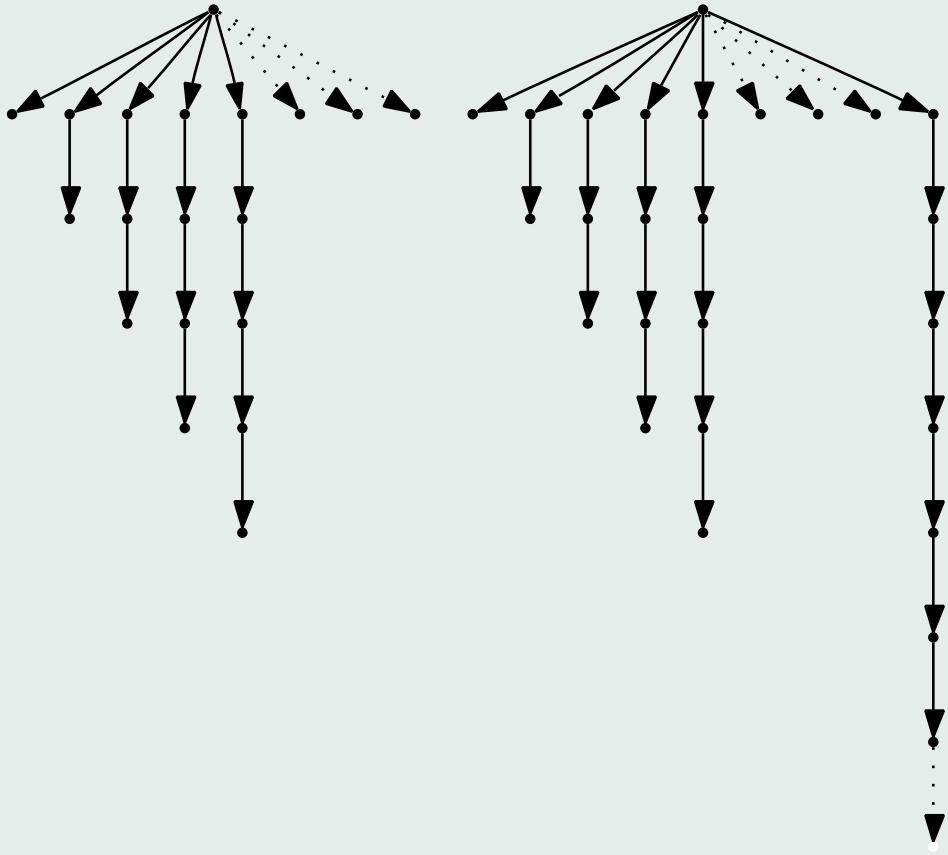
What Theorem 1 says is that $\mathbf{M}, s \underline{\leftrightarrow} \mathbf{N}, t$ implies that $\mathbf{M}, s \leftrightsquigarrow \mathbf{N}, t$.

But does the converse also hold?

Does it follow from $\mathbf{M}, s \leftrightsquigarrow \mathbf{N}, t$ that $\mathbf{M}, s \underline{\leftrightarrow} \mathbf{N}, t$?

The answer is no.

Or maybe a better answer is: almost.



This example is not image-finite.

In both models, the root state has an infinite number of successors.

A Kripke model is image-finite if it holds for every world w and every accessibility relation R in the model that

$$\{v \mid wRv\}$$

(the image of w under R) is a finite set.

Theorem 3 (Hennessy-Milner) *If* $\mathbf{M}$ *and* $\mathbf{N}$ *are image-finite then* $\mathbf{M}, s \leftrightsquigarrow \mathbf{N}, t$ *implies* $\mathbf{M}, s \underline{\leftrightarrow} \mathbf{N}, t$.

Proof: We will show that the relation $\leftrightsquigarrow$ is itself a bisimulation. For that, we have to demonstrate that $\leftrightsquigarrow$ satisfies the invariance, zig and zag conditions.

Invariance. Assume $s \leftrightsquigarrow t$. Then surely s and t have the same valuation, for otherwise even a propositional formula would see a difference.

Zig. Assume $s \leftrightsquigarrow t$, and let $s \xrightarrow{a} s'$. We have to show that there is a t' with $t \xrightarrow{a} t'$ and $s' \leftrightsquigarrow t'$.

For a contradiction, assume that there is no such t'.

By the fact that $s \xrightarrow{a} s'$ the formula $\langle a \rangle \top$ holds at s. By $s \leftrightsquigarrow t$, formula $\langle a \rangle \top$ also holds at t.

Therefore the set $U = \{u \mid t \xrightarrow{a} u\}$ is non-empty.

Since $\mathbf{N}$ is image-finite, U must be finite. Let us say

$$U = \{u_1, \ldots, u_n\}.$$

Now for every $u_i \in U$ there must be a ψ_i with $\mathbf{M}, s' \models \psi_i$ and $\mathbf{N}, u_i \not\models \psi_i$. But from this it follows that:

$$\mathbf{M}, s \models \langle a \rangle(\psi_1 \wedge \cdots \wedge \psi_n) \text{ and } \mathbf{N}, t \not\models \langle a \rangle(\psi_1 \wedge \cdots \wedge \psi_n).$$

This contradicts the given that $s \leftrightsquigarrow t$.

Thus there is a t' with $t \xrightarrow{a} t'$ and $s' \leftrightsquigarrow t'$.

Zag. Same reasoning, but now in the other direction, using the image-finiteness of $\mathbf{M}$.

Bisimilarity is an Equivalence

We consider bisimilarity within a single model $\mathbf{M}$, and we show that this is an equivalence relation.

Reflexive Surely $I : s \underline{\leftrightarrow} s$ for every state s in $\mathbf{M}$ (the identity relation is a bisimulation). Thus $s \underline{\leftrightarrow} s$.

Symmetric Let $s \underline{\leftrightarrow} t$. Then there is a Z with $Z : s \underline{\leftrightarrow} t$. Note that the invariance, zig, and zag conditions are symmetric. Thus, $Z : s \underline{\leftrightarrow} t$ implies $Z^{\smile} : t \underline{\leftrightarrow} s$. Thus, $t \underline{\leftrightarrow} s$.

Transitive Assume $s \underline{\leftrightarrow} t$ and $t \underline{\leftrightarrow} u$. Then there are $Z : s \underline{\leftrightarrow} t$ and $Z' : t \underline{\leftrightarrow} u$. But then $Z \circ Z' : s \underline{\leftrightarrow} u$, and therefore $s \underline{\leftrightarrow} u$.

It follows that we can simplify a Kripke model by replacing each state s by its bisimilarity-class $|s|_{\underline{\leftrightarrow}}$.

By Theorem 1 this does not affect the truth of any modal formulas.

Partition Refinement

Given: A Kripke model $\mathbf{M}$.

Problem: find the Kripke model that results from replacing each state s in $\mathbf{M}$ by its bisimilarity class $|s|_{\underline{\leftrightarrow}}$.

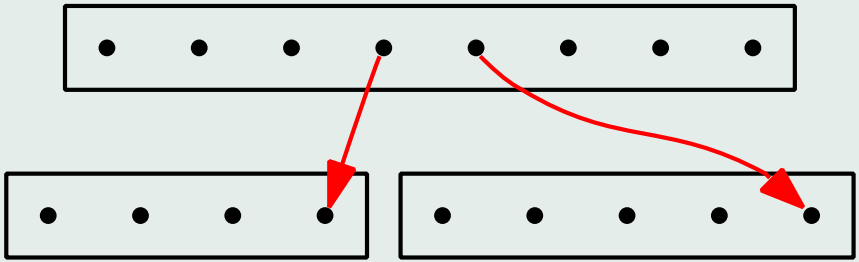
The problem of finding the smallest Kripke model modulo bisimulation is similar to the problem of minimizing the number of states in a finite automaton [3].

We will use partition refinement, in the spirit of [4].

Partition Refinement Algorithm

- Start out with a partition of the state set where all states with the same valuation are in the same class.
- Given a partition Π, for each block b in Π, partition b into sub-blocks such that two states s, t of b are in the same sub-block iff for all agents a it holds that s and t have $\xrightarrow{a}$ transitions to states in the same block of Π. Update Π to Π' by replacing each b in Π by the newly found set of sub-blocks for b.
- Halt as soon as $\Pi = \Pi'$.

Splitting a Block



Module Declaration

```
module LAI11 where

import List
import Char
import LAI9
import LAI10
type State = Integer
```

Valuation Comparison

```
sameVal :: (Eq a,Eq b) => [(a,b)] -> a -> a -> Bool
sameVal val w1 w2 = apply val w1 == apply val w2
```

From Equivalence Relations to Partitions

```
rel2part :: (Eq a) =>
             [a] -> (a -> a -> Bool) -> [[a]]
rel2part [] r = []
rel2part (x:xs) r = xblock : rel2part rest r
  where
  (xblock,rest) = (x:filter (r x) xs,
                   filter (not . (r x)) xs)
```

Initial Partition

We start with the partition based on the relation 'having the same valuation':

```
initPartition :: Eq a => EpistM a -> [[a]]
initPartition (Mo states agents val rel actual) =
  rel2part states (\ x y -> sameVal val x y)
```

The block of an object in a partition

The block of `x` in a partition is the block that has `x` as an element.

```
bl :: Eq a => [[a]] -> a -> [a]
bl part x = head (filter (elem x) part)
```

Accessible Blocks

For an agent from a given state, given a model and a partition:

```
accBlocks :: Eq a =>
             EpistM a -> [[a]] -> a -> Agent -> [[a]]
accBlocks m@(Mo _ _ _ rel _) part s ag =
    nub [ bl part y | (ag',x,y) <- rel,
                      ag' == ag, x == s ]
```

Having the same accessible blocks under a partition

```
sameAB :: Eq a =>
          EpistM a -> [[a]] -> a -> a -> Bool
sameAB m@(Mo states ags val rel actual) part s t =
    and [ accBlocks m part s ag
          == accBlocks m part t  ag | ag <- ags ]
```

Refinement Step of Partition by Block Splitting

Splitting the blocks `bl` of `p`:

```
refineStep :: Eq a => EpistM a -> [[a]] -> [[a]]
refineStep m p = refineP m p p
  where
  refineP :: Eq a =>
             EpistM a -> [[a]] -> [[a]] -> [[a]]
  refineP m part [] = []
  refineP m part (bl:blocks) =
     newblocks ++ (refineP m part blocks)
       where
         newblocks =
          rel2part bl (\ x y -> sameAB m part x y)
```

Refining a Partition

The refining process can be implemented as a least fixpoint computation on the operation of taking refinement steps.

```
refine :: Eq a => EpistM a -> [[a]] -> [[a]]
refine m = lfp (refineStep m)
```

Remark: least fixpoint computation is an element of many refinement processes. It is an example of what is called a design pattern in Software Engineering [2].

Construction of Minimal Model

```
minimalModel :: (Eq a, Ord a) =>
                 EpistM a -> EpistM [a]
minimalModel m@(Mo states agents val rel actual) =
  (Mo states' agents val' rel' actual')
     where
     states'   = refine m (initPartition m)
     f         = bl states'
     val'      = (nub . sort)
        (map (\ (x,y)   -> (f x, y))      val)
     rel'      = (nub . sort)
        (map (\ (x,y,z) -> (x, f y, f z)) rel)
     actual'   = map f actual
```

Map to Bisimulation Minimal Model

Map the states to their bisimilarity classes.

Next, convert the bisimilarity classes back into integers:

```
bisim ::  (Eq a, Ord a) =>
           EpistM a -> EpistM State
bisim = convert . minimalModel
```

Examples

```
lai0 :: EpistM State
lai0 = Mo
         [0..7]
         [a,b,c]
         (zip [0..]
          ((powerList [P 1, P 2])
            ++ (powerList [P 1, P 2])))
         [(ag,x,x) | ag <- [a,b,c], x <- [0..7] ]
         [2]
```

```
LAI11> displayS5 lai0
[0,1,2,3,4,5,6,7]
[(0,[]),(1,[p2]),(2,[p1]),(3,[p1,p2]),(4,[]),
 (5,[p2]),(6,[p1]),(7,[p1,p2])]
(a,[[0],[1],[2],[3],[4],[5],[6],[7]])
(b,[[0],[1],[2],[3],[4],[5],[6],[7]])
(c,[[0],[1],[2],[3],[4],[5],[6],[7]])
[2]

LAI11> displayS5 (bisim lai0)
[0,1,2,3]
[(0,[]),(1,[p2]),(2,[p1]),(3,[p1,p2])]
(a,[[0],[1],[2],[3]])
(b,[[0],[1],[2],[3]])
(c,[[0],[1],[2],[3]])
[2]
```

```
lai1 :: EpistM State
lai1 = let worlds = [0..10] in
       Mo
        worlds
        [a,b,c]
        (zip worlds (repeat [P 0]))
        [(ag,x,y) | ag <- [a,b,c],
                    x  <- worlds, y <- worlds ]
        [10]
```

```
LAI11> displayS5 lai1
[0,1,2,3,4,5,6,7,8,9,10]
[(0,[p]),(1,[p]),(2,[p]),(3,[p]),(4,[p]),(5,[p]),
 (6,[p]),(7,[p]),(8,[p]),(9,[p]),(10,[p])]
(a,[[0,1,2,3,4,5,6,7,8,9,10]])
(b,[[0,1,2,3,4,5,6,7,8,9,10]])
(c,[[0,1,2,3,4,5,6,7,8,9,10]])
[10]

LAI11> displayS5 (bisim lai1)
[0]
[(0,[p])]
(a,[[0]])
(b,[[0]])
(c,[[0]])
[0]
```

```
lai2 :: EpistM State
lai2 = let worlds = [0..10] in
       Mo
        worlds
        [a,b,c]
        ((0,[Q 0]): (zip [1..10] (repeat [P 0])))
        [(ag,x,y) | ag <- [a,b,c],
                    x  <- worlds, y <- worlds ]
        [10]
```

```
LAI11> displayS5 lai2
[0,1,2,3,4,5,6,7,8,9,10]
[(0,[q]),(1,[p]),(2,[p]),(3,[p]),(4,[p]),(5,[p]),
 (6,[p]),(7,[p]),(8,[p]),(9,[p]),(10,[p])]
(a,[[0,1,2,3,4,5,6,7,8,9,10]])
(b,[[0,1,2,3,4,5,6,7,8,9,10]])
(c,[[0,1,2,3,4,5,6,7,8,9,10]])
[10]

LAI11> displayS5 (bisim lai2)
[0,1]
[(0,[q]),(1,[p])]
(a,[[0,1]])
(b,[[0,1]])
(c,[[0,1]])
[1]
```

Next Time

- Common knowledge and public announcement again.
- Other update functions.
- Action models.
- Updating with an action model.

References

[1] P. Blackburn, M. de Rijke, and Y. Venema. Modal Logic. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

[2] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley Professional, 1995.

[3] J.E.Hopcroft. An n log n algorithm for minimizing states in a finite automaton. In Zvi Kohavi and Azaria Paz, editors, Theory of Machines and Computations. Academic Press, 1971.

[4] Robert Paige and Robert E. Tarjan. Three partition refinement algorithms. SIAM J. Comput., 16(6):973–989, 1987.