AI Planning: Planning with Dynamic Epistemic Logic

Markus Eger

2019-11-15

Outline

Outline

Review: The Planning Problem

A planning problem consists of three parts:

- A definition of the current state of the world
- A definition of a desired state of the world
- A definition of the actions the agent can take

Outline

One Night Ultimate Werewolf

(Source: whatsericplaying.com)

One Night Ultimate Werewolf

- Competitive
- Players are secretly assigned roles
- Factions: Werewolves, Villagers
- Goal of the Villagers: Find the Werewolves
- Goal of the Werewolves: Avoid detection
- Night phase for actions, day phase for communication
- Players can lie
- Players' roles can change without their knowledge

One Night Ultimate Werewolf - Roles

- Seer: Look at another player's card

One Night Ultimate Werewolf - Roles

- Seer: Look at another player's card
- Robber: Exchange cards with another player, look at the new card

One Night Ultimate Werewolf - Roles

- Seer: Look at another player's card
- Robber: Exchange cards with another player, look at the new card
- Rascal: May exchange your two neighbors' cards

One Night Ultimate Werewolf - Roles

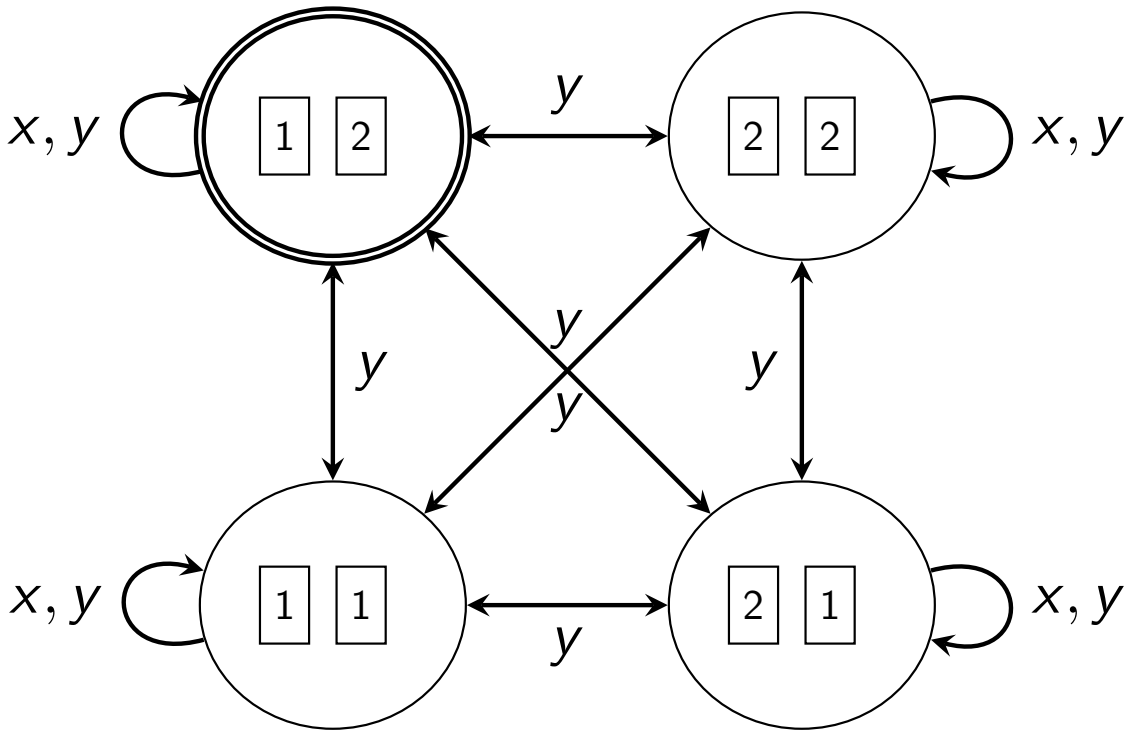
- Seer: Look at another player's card
- Robber: Exchange cards with another player, look at the new card
- Rascal: May exchange your two neighbors' cards
- Insomniac: Look at your own card

One Night Ultimate Werewolf - Challenges

- Hidden information
- Deception
- Suspicion
- Good game play requires a model of belief

Outline

Formal Model: Possible Worlds

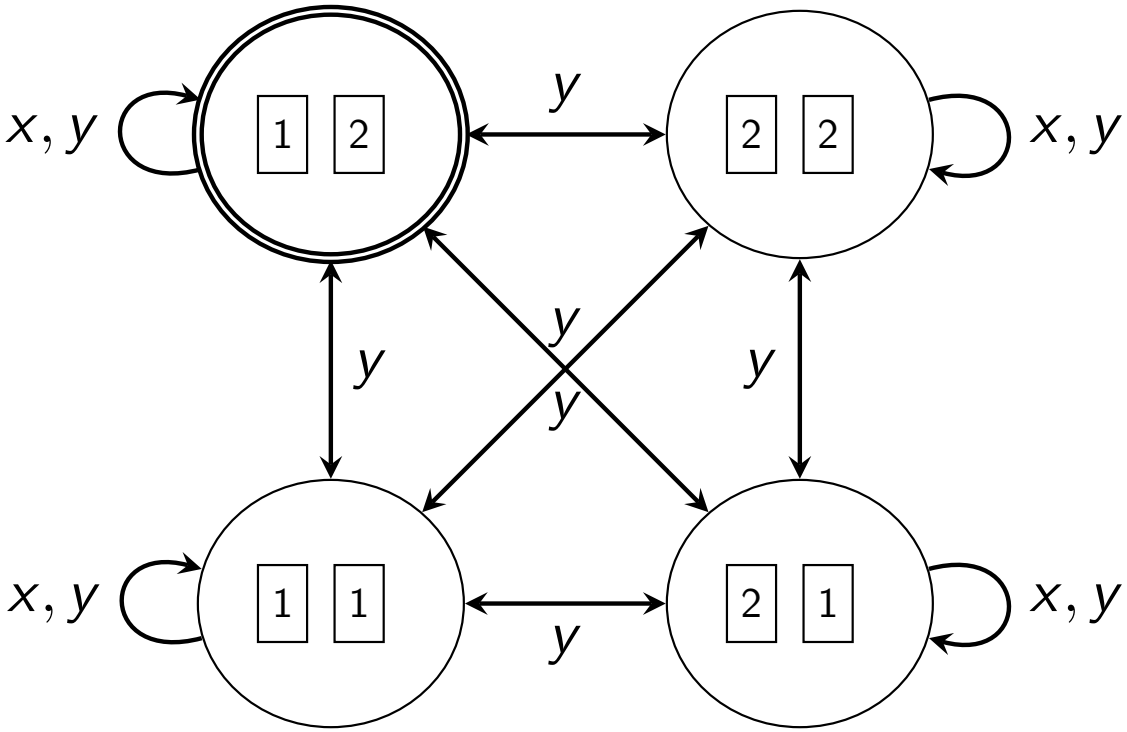


Formal Model: Dynamic Epistemic Logic

- Baltag's variant of Dynamic Epistemic Logic
- Possible worlds represent agents' beliefs
- Actions can add worlds (increase uncertainty) or remove worlds (decrease uncertainty)

Dynamic Epistemic Logic - Possible Worlds

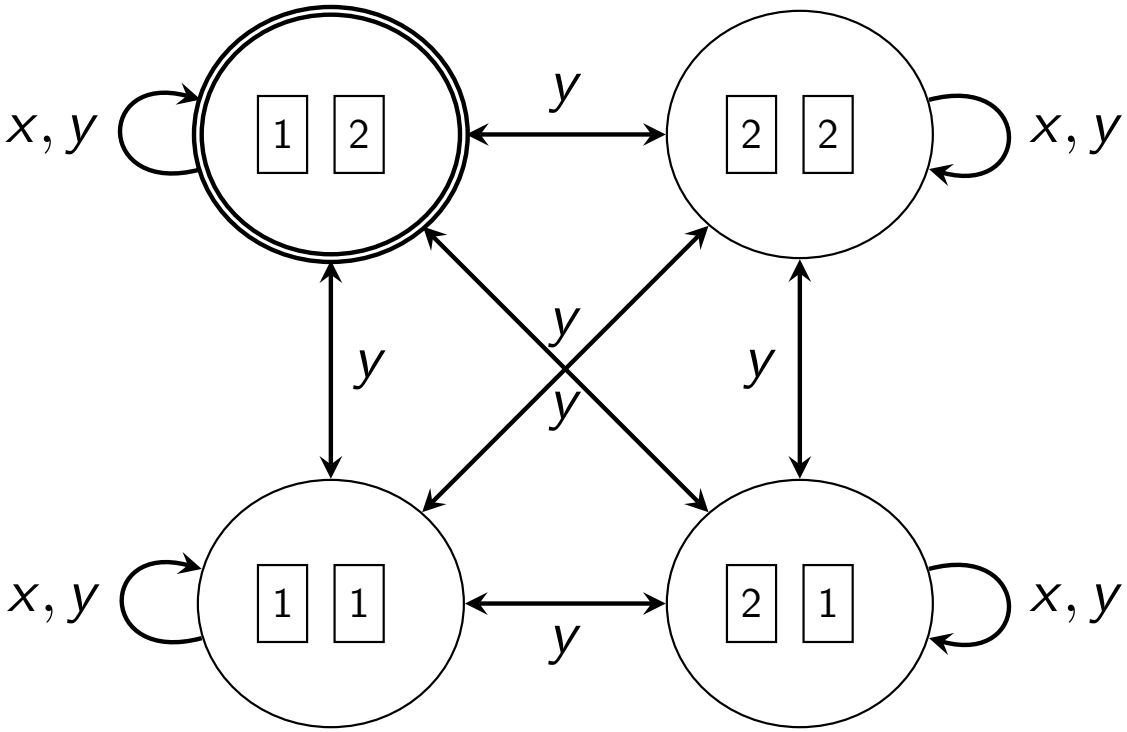
```
learn (y): left(x) == 1
```



Dynamic Epistemic Logic - Possible Worlds

```
learn (y): left(x) == 1
```

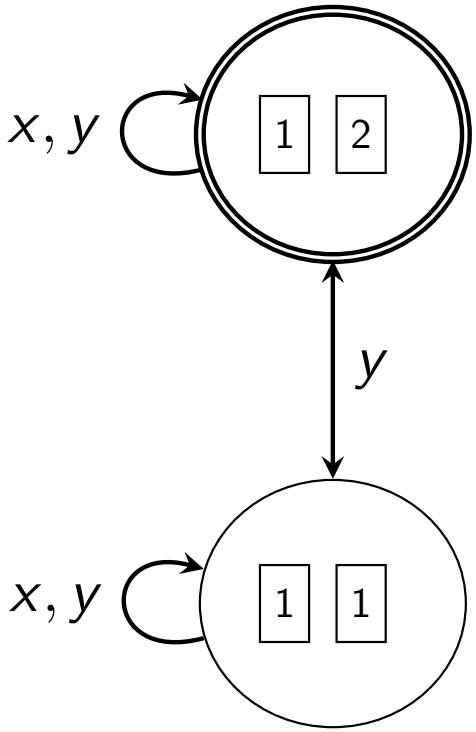
$(?\,\mathrm{left}(x, 1))^{*y}$



Dynamic Epistemic Logic - Possible Worlds

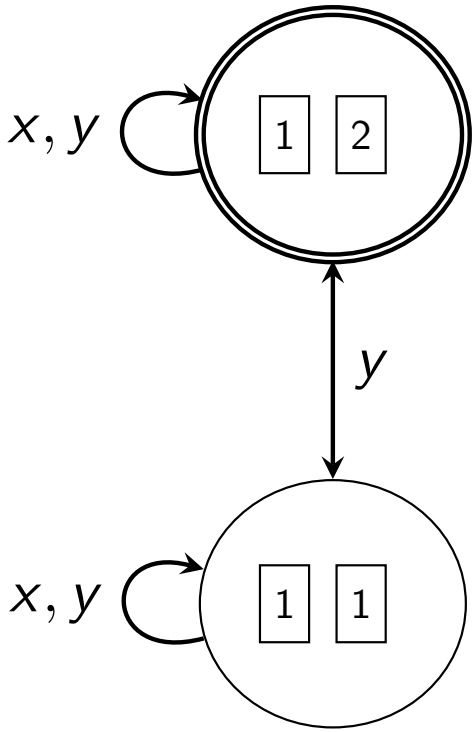
```
learn (y): left(x) == 1
```

$(?\,\mathrm{left}(x, 1))^{*y}$



Dynamic Epistemic Logic - Possible Worlds

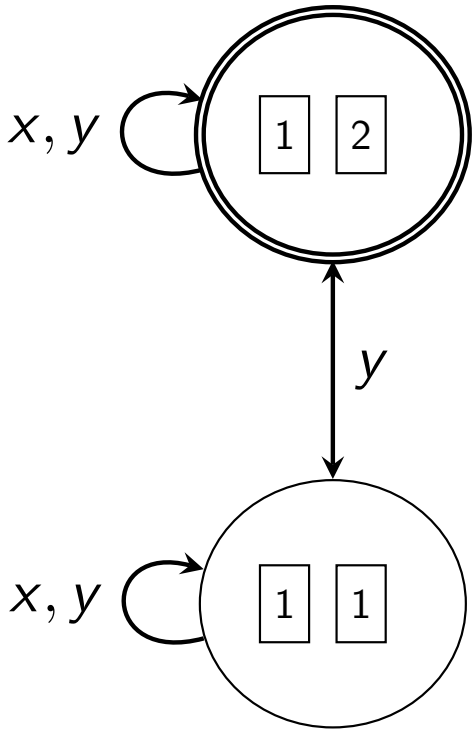
```
drawright(p: Players, c(p): Cards)
    right(p) = c
```



Dynamic Epistemic Logic - Possible Worlds

```
drawright(p: Players, c(p): Cards)
    right(p) = c
```

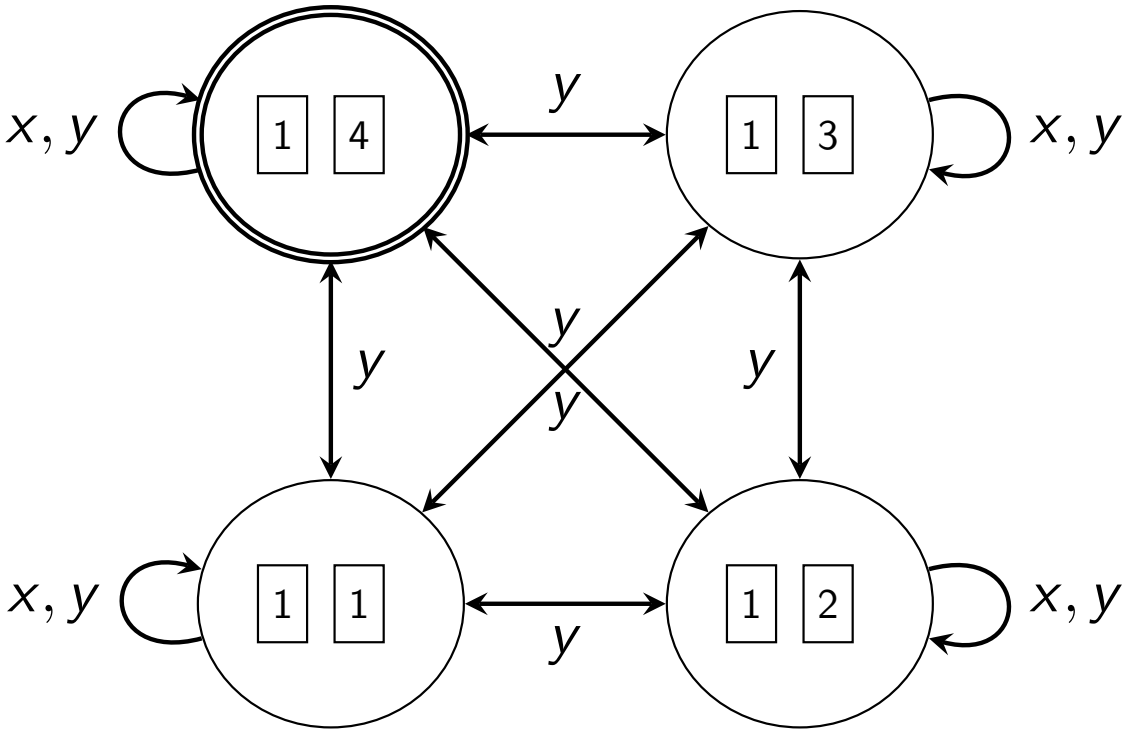
$(\text{flip right}(x, 2) \cdot \text{flip right}(x, 4))^{*x} \cdot$
$(\text{flip right}(x, 2) \cdot (\text{flip right}(x, 1) + \text{flip right}(x, 2) + \text{flip right}(x, 3) +$
$\text{flip right}(x, 4)))^{y}$



Dynamic Epistemic Logic - Possible Worlds

```
drawright(p: Players, c(p): Cards)
    right(p) = c
```

$(\text{flip right}(x, 2) \cdot \text{flip right}(x, 4))^{*x} \cdot$
$(\text{flip right}(x, 2) \cdot (\text{flip right}(x, 1) + \text{flip right}(x, 2) + \text{flip right}(x, 3) +$
$\text{flip right}(x, 4)))^{y}$



Dynamic Epistemic Logic - Limitations

- Basic operation flip changes one bit at a time
- Game state has to be encoded in one bit units
- Actions have to change single bits at a time
- Interesting actions are cumbersome to write

Example: Swap two players' roles

```
(?role(b,Rascal) . ?phase(game,TMPhase) .
(?role(tmp,Werewolf) . flip role(tmp,Werewolf) + ?role(tmp,Rascal) . flip role(tmp,Rascal) +
 ?role(tmp,Villager) . flip role(tmp,Villager) + ?!role(tmp,Werewolf) .
 ?!role(tmp,Rascal) . ?!role(tmp,Villager)) . (?role(a,Werewolf) . flip role(tmp,Werewolf) +
 ?role(a,Rascal) . flip role(tmp,Rascal) + ?role(a,Villager) . flip role(tmp,Villager)) .
 (?role(a,Werewolf) . flip role(a,Werewolf) + ?role(a,Rascal) . flip role(a,Rascal) +
 ?role(a,Villager) . flip role(a,Villager) + ?!role(a,Werewolf) . ?!role(a,Rascal) . ?!role(a,Villager)) .
 (?role(c,Werewolf) . flip role(a,Werewolf) + ?role(c,Rascal) . flip role(a,Rascal) +
 ?role(c,Villager) . flip role(a,Villager)) . (?role(c,Werewolf) . flip role(c,Werewolf) + ?role(c,Rascal) .
 flip role(c,Rascal) + ?role(c,Villager) . flip role(c,Villager) + ?!role(c,Werewolf) . ?!role(c,Rascal) .
 ?!role(c,Villager)) . (?role(tmp,Werewolf) . flip role(c,Werewolf) + ?role(tmp,Rascal) .
 flip role(c,Rascal) + ?role(tmp,Villager) . flip role(c,Villager)) . (?role(b,Rascal) .
 ?phase(game,TMPhase) . (?role(tmp,Werewolf) . flip role(tmp,Werewolf) + ?role(tmp,Rascal) .
 flip role(tmp,Rascal) + ?role(tmp,Villager) . flip role(tmp,Villager) + ?!role(tmp,Werewolf) .
 ?!role(tmp,Rascal) . ?!role(tmp,Villager)) . (?role(a,Werewolf) . flip role(tmp,Werewolf) +
 ?role(a,Rascal) . flip role(tmp,Rascal) + ?role(a,Villager) . flip role(tmp,Villager)) .
 (?role(a,Werewolf) . flip role(a,Werewolf) + ?role(a,Rascal) . flip role(a,Rascal) +
 ?role(a,Villager) . flip role(a,Villager) + ?!role(a,Werewolf) . ?!role(a,Rascal) . ?!role(a,Villager)) .
 (?role(a,Werewolf) . flip role(a,Werewolf) + ?role(a,Rascal) . flip role(a,Rascal) + ?role(a,Villager) .
 flip role(a,Villager)) . (?role(a,Werewolf) . flip role(a,Werewolf) + ?role(a,Rascal) . flip role(a,Rascal) +
 ?role(a,Villager) . flip role(a,Villager) + ?!role(a,Werewolf) . ?!role(a,Rascal) . ?!role(a,Villager)) .
 (?role(tmp,Werewolf) . flip role(a,Werewolf) + ?role(tmp,Rascal) . flip role(a,Rascal) +
 ?role(tmp,Villager) . flip role(a,Villager)) + ?role(b,Rascal) . ?phase(game,TMPhase) .
 (?role(tmp,Werewolf) . flip role(tmp,Werewolf) + ?role(tmp,Rascal) . flip role(tmp,Rascal) +
 ?role(tmp,Villager) . flip role(tmp,Villager) + ?!role(tmp,Werewolf) . ?!role(tmp,Rascal) . ?!role(tmp,Village
 (?role(a,Werewolf) . flip role(tmp,Werewolf) + ?role(a,Rascal) . flip role(tmp,Rascal) +
 ?role(a,Villager) . flip role(tmp,Villager)) . (?role(a,Werewolf) . flip role(a,Werewolf) + ?role(a,Rascal) .
 flip role(a,Rascal) + ?role(a,Villager) . flip role(a,Villager) + ?!role(a,Werewolf) . ?!role(a,Rascal) .
 ?!role(a,Villager)) . (?role(b,Werewolf) . flip role(a,Werewolf) + ?role(b,Rascal) . flip role(a,Rascal) +
 ?role(b,Villager) . flip role(a,Villager)) . (?role(b,Werewolf) . flip role(b,Werewolf) + ?role(b,Rascal) .
 flip role(b,Rascal) + ?role(b,Villager) . flip role(b,Villager) + ?!role(b,Werewolf) . ?!role(b,Rascal) .
```

Example: Swap two players' roles

Example: Swap two players' roles

```
causeRuckus(p(p): Players, t(p): Bool)
{
    precondition initrole(p) == Rascal;
    if (eqt(t) == True)
    {
        role(tmp) = role(left(p));
        role(left(p)) = role(right(p));
        role(right(p)) = role(tmp);
    }
    else
    {

    }
}
```

Outline

Planning with Dynamic Epistemic Logic

- Define initial epistemic state
- Define epistemic actions
- Define a goal condition
- Task: Find a sequence of actions such that the goal is satisfied

Planning with Dynamic Epistemic Logic

- Define initial epistemic state
- Define epistemic actions
- Define a goal condition
- Task: Find a sequence of actions such that the goal is satisfied(-ish)

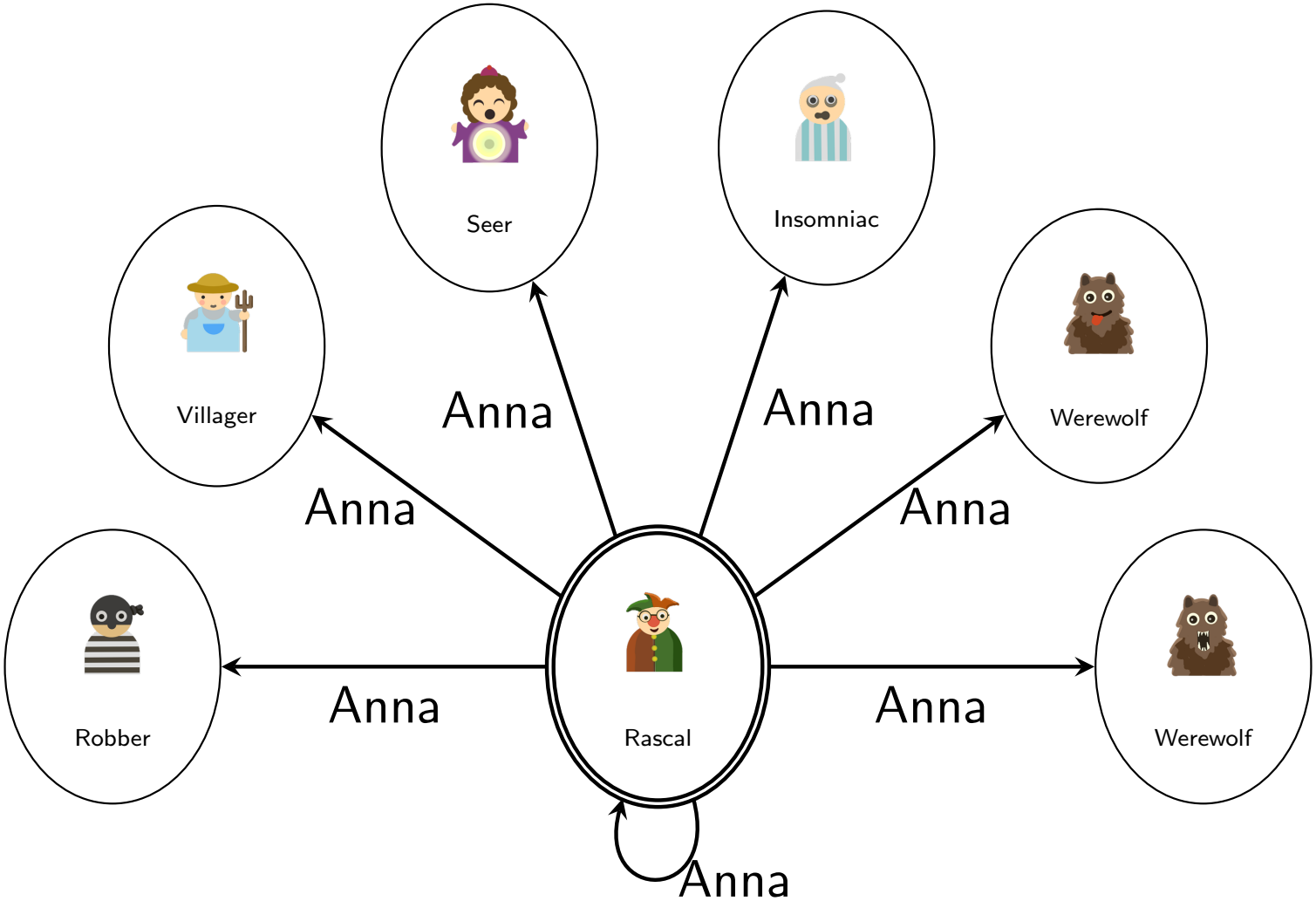
Planning with Dynamic Epistemic Logic

- Define initial epistemic state
- Define epistemic actions
- Define a goal condition
- Task: Find a sequence of actions such that the goal is satisfied(-ish)
- What if the goal can not be reached?

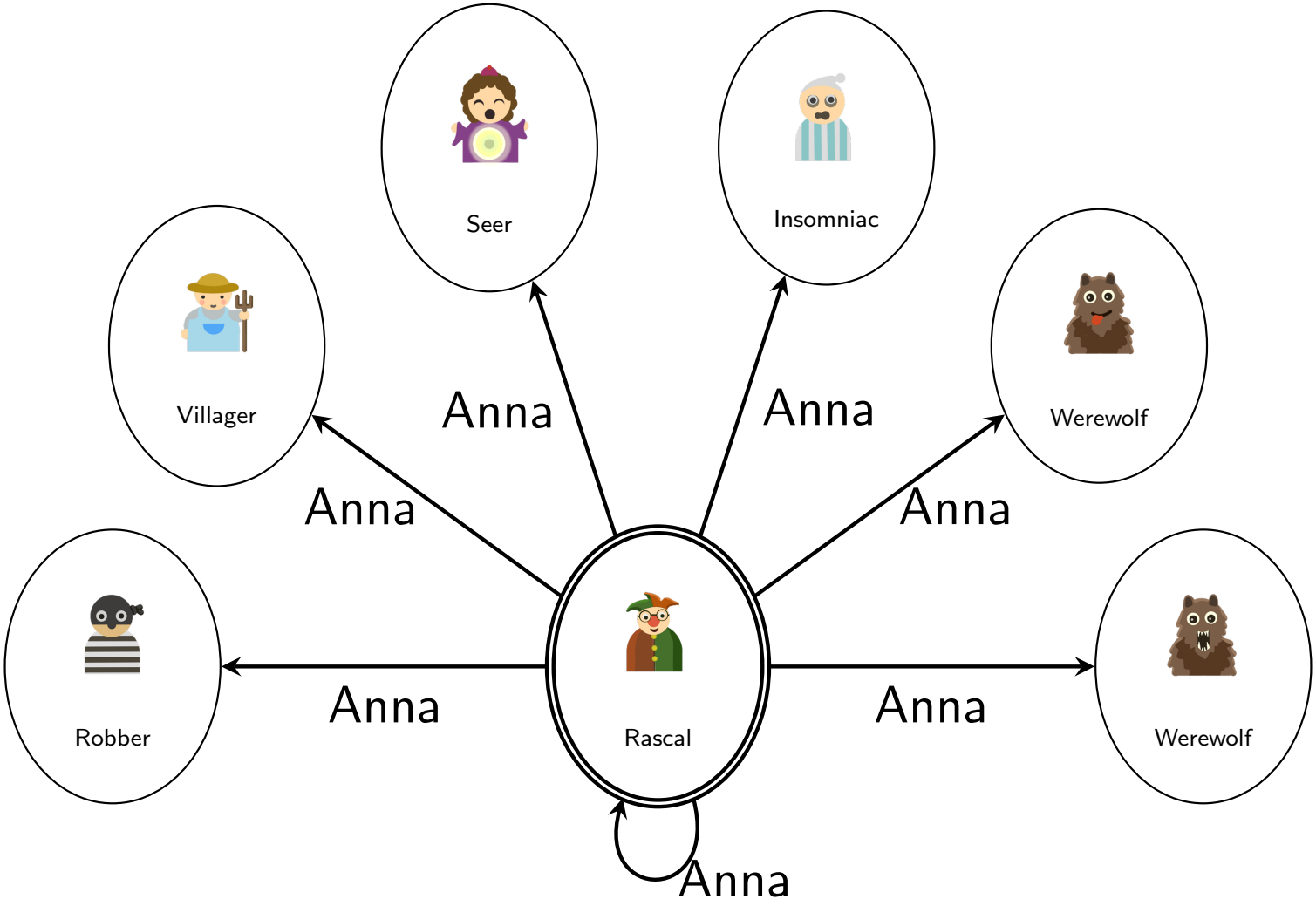
A simple scenario

- Anna and Brian play One Night Ultimate Werewolf
- Anna got a Villager card
- Brian got the Rascal card
- But Anna does not know which card Brian has

Belief Quality

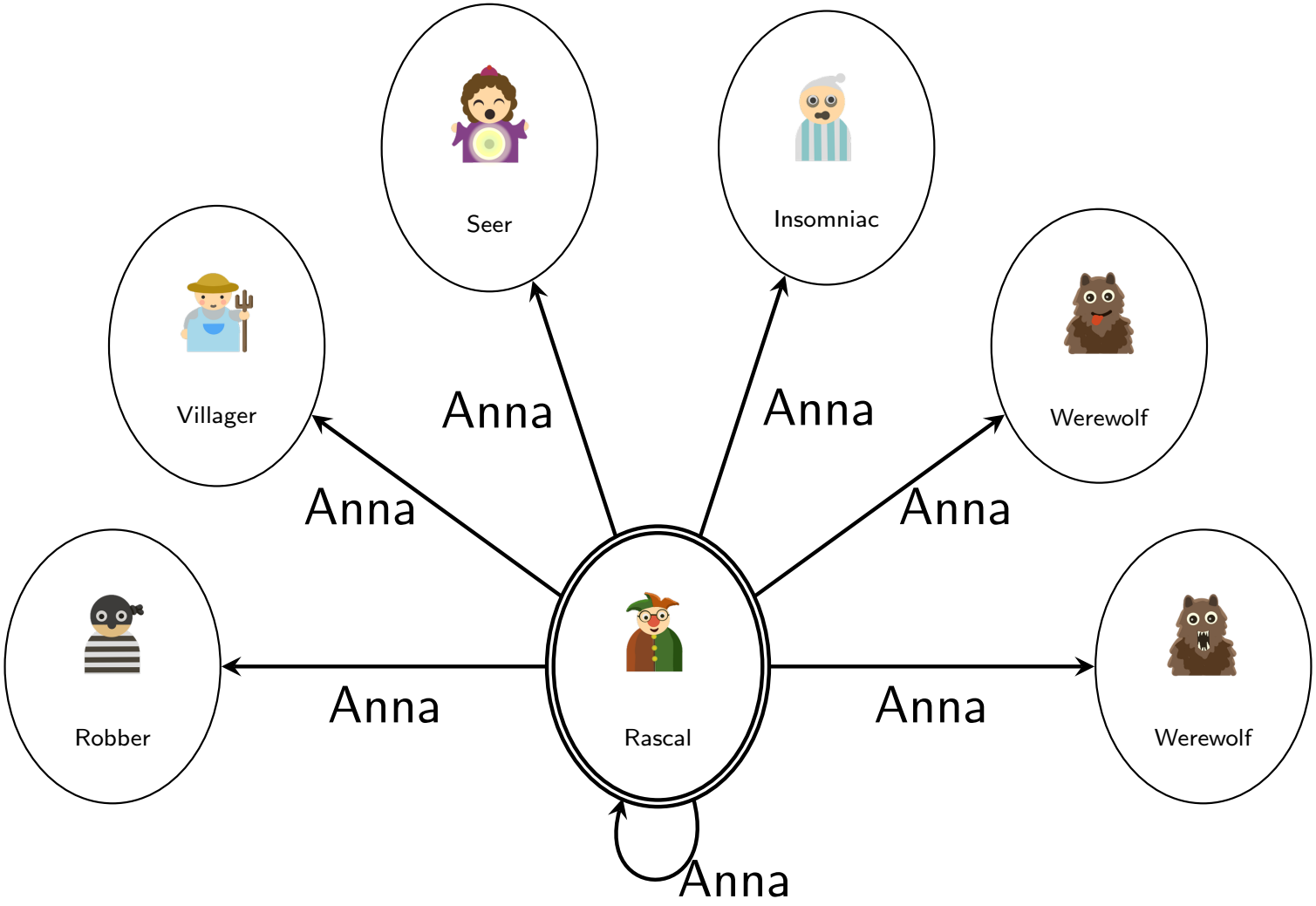


Belief Quality



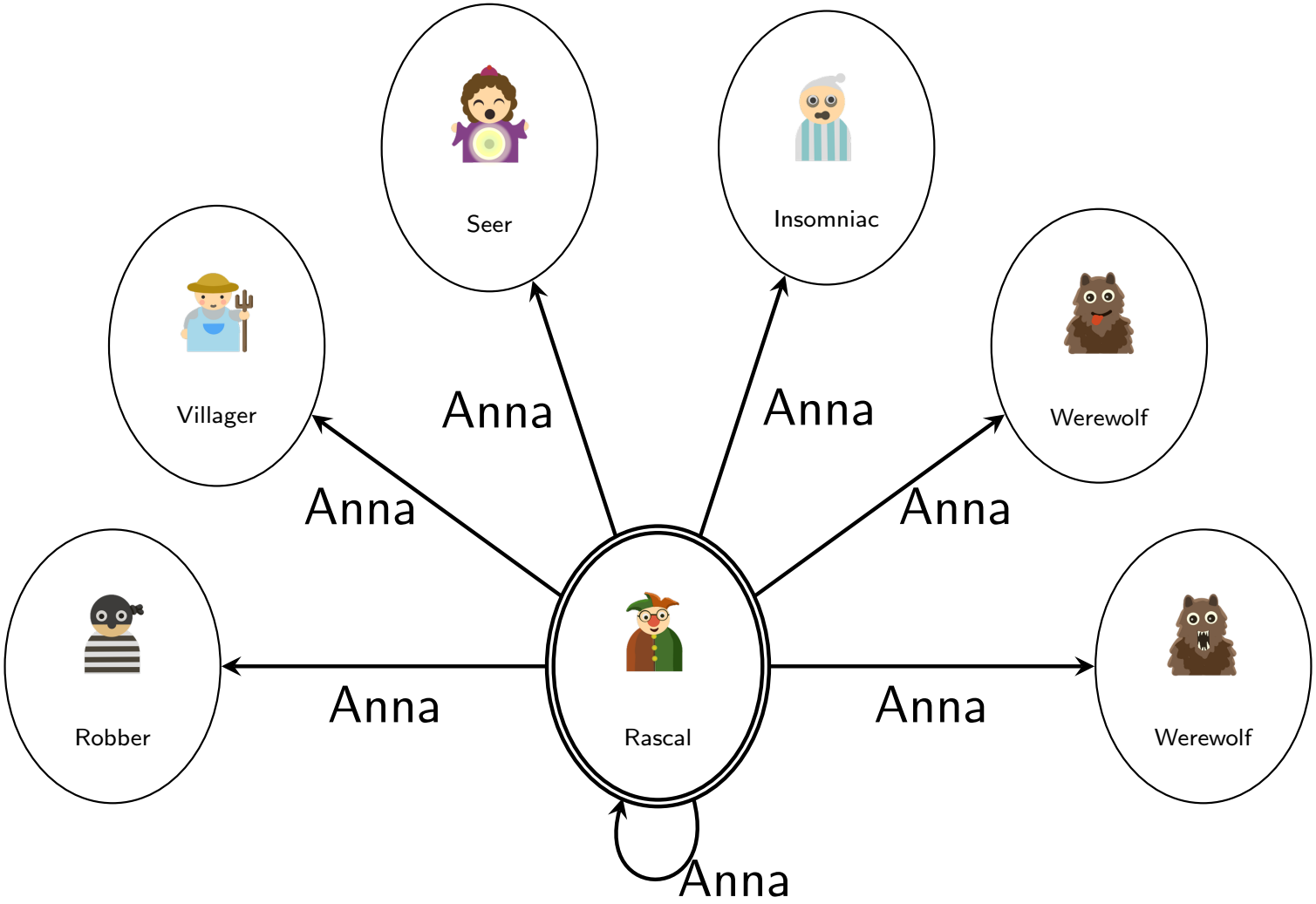
$$\mathcal{M} \not\models \Box_{Anna}\, \mathrm{role}(Brian) = Werewolf$$

Belief Quality



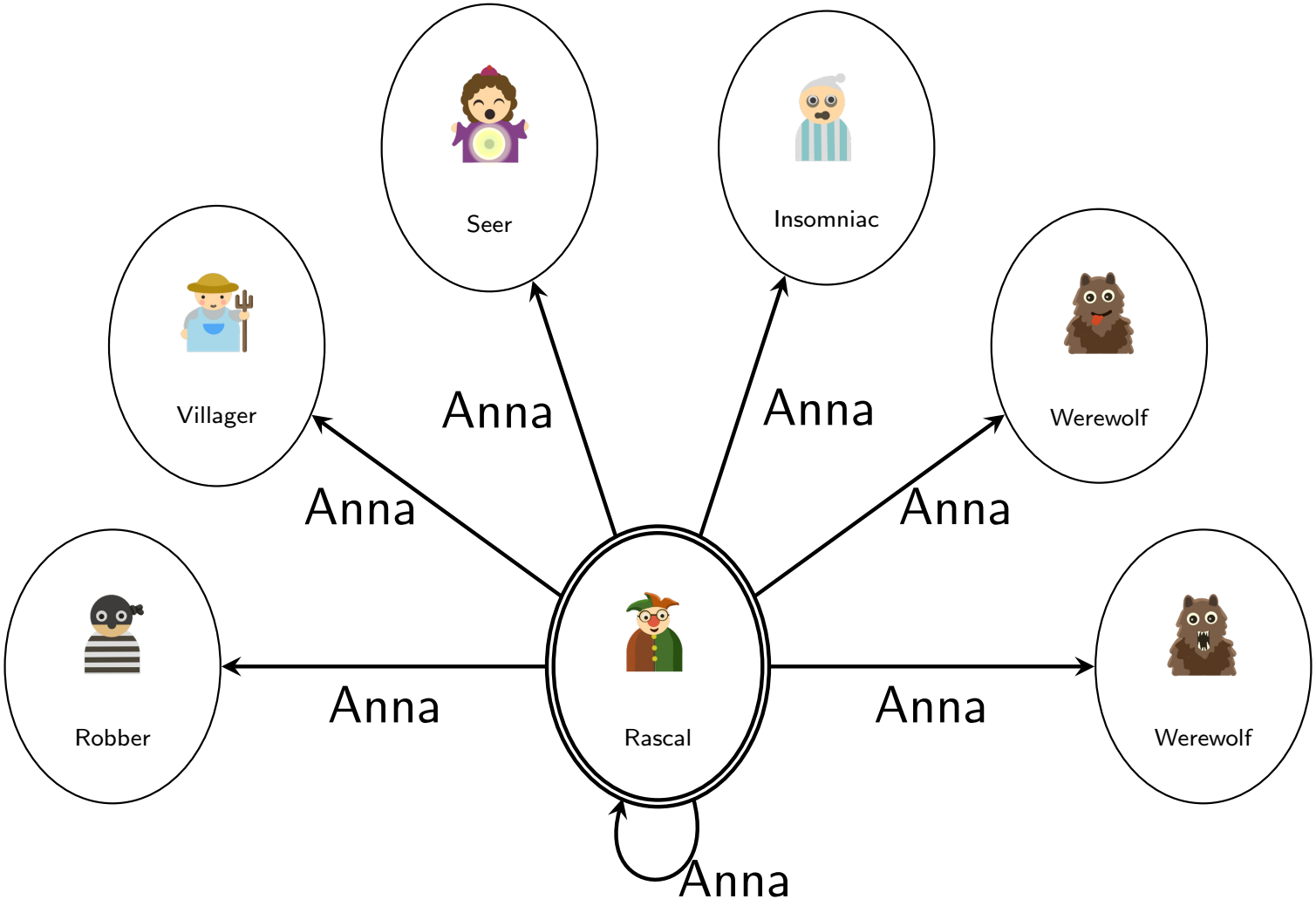
$$\mathcal{M} \not\models \Box_{Anna}\, \mathrm{role}(Brian) \neq Werewolf$$

Belief Quality



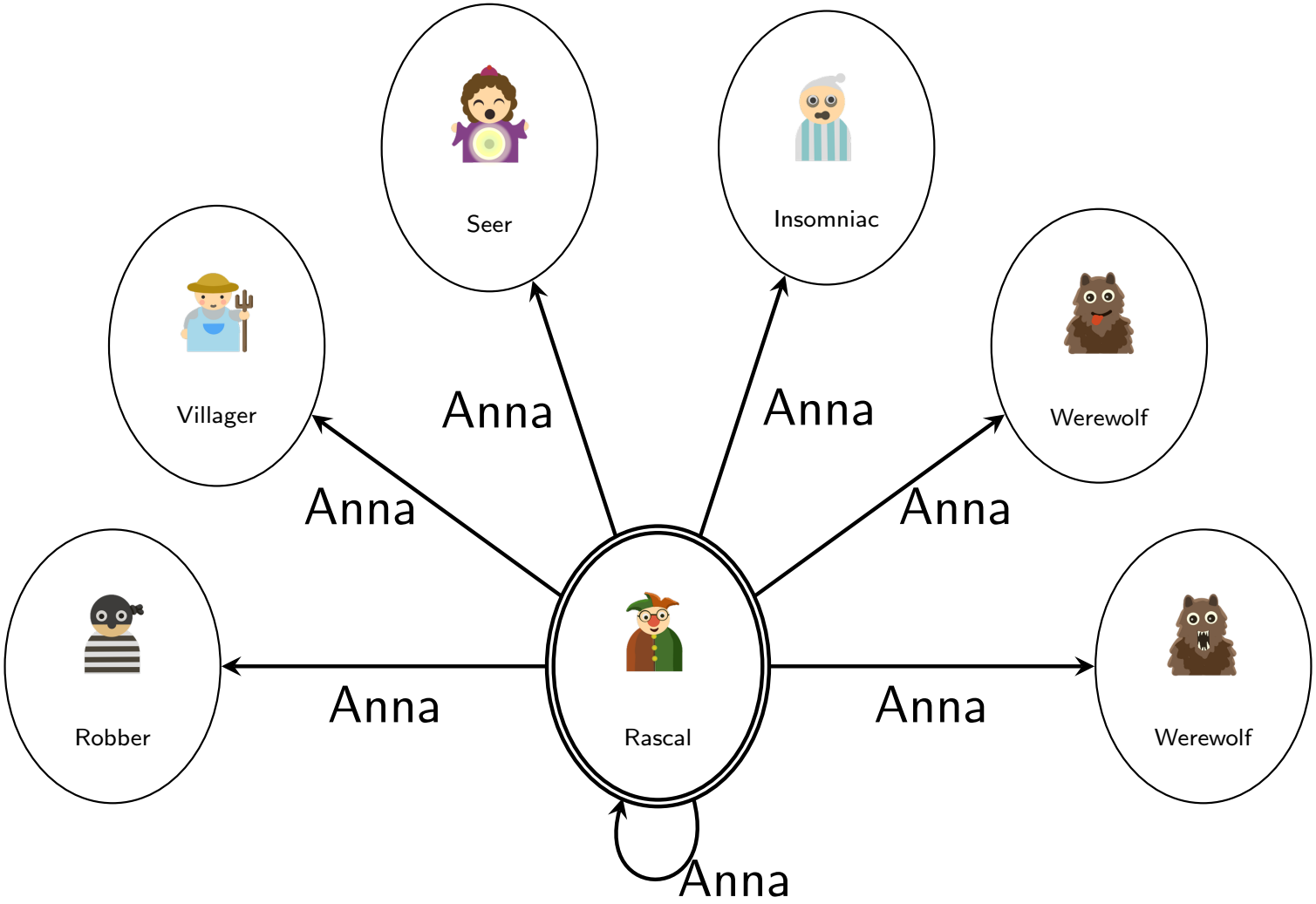
Brian is a Werewolf in 2 out of 7 worlds!

Belief Quality



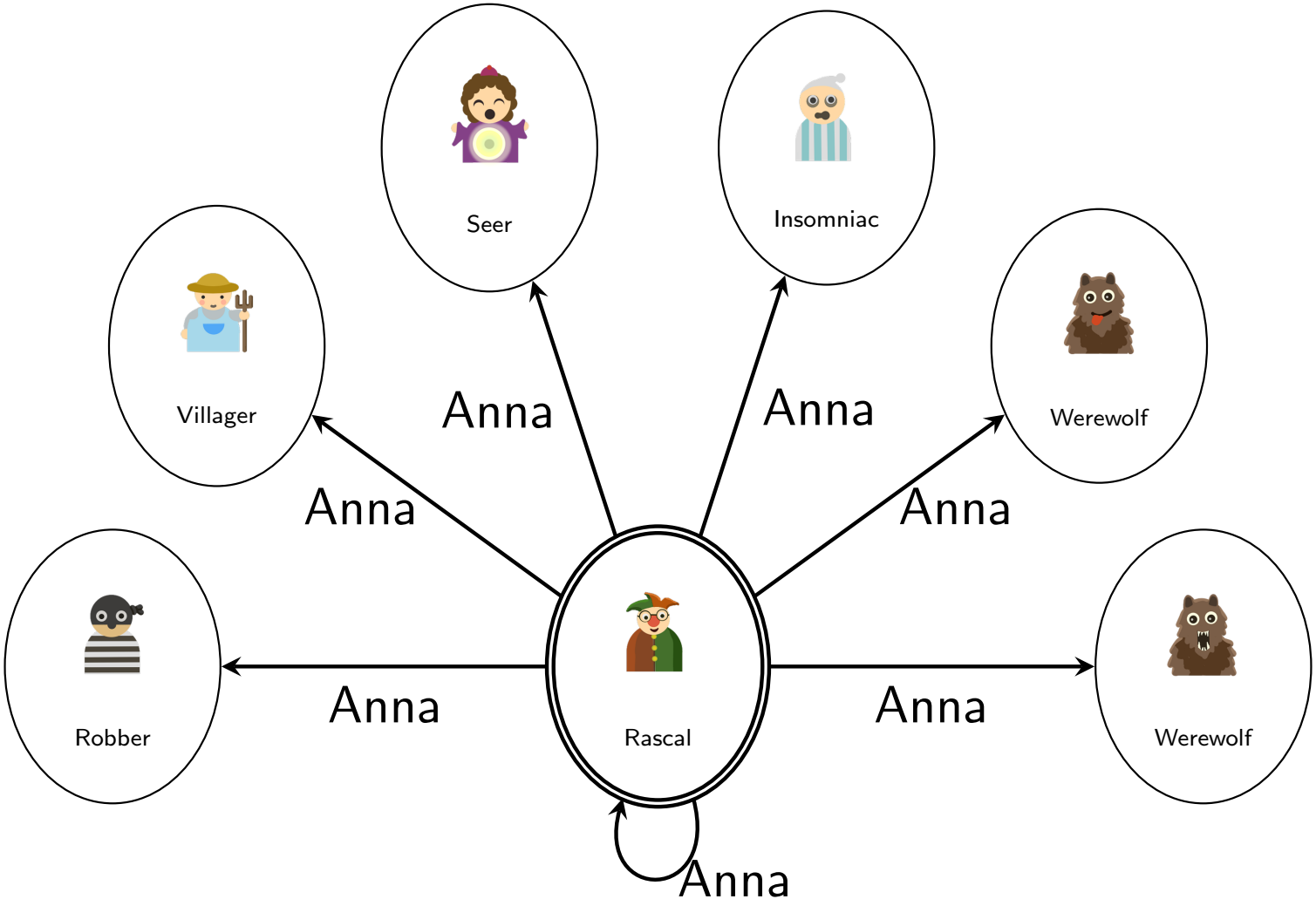
$$\mathcal{M} \models Q[\tfrac{2}{7}]_{Anna}\,\mathrm{role}(Brian) = Werewolf$$

Belief Quality



$$\mathcal{M} \models Q[\geq \tfrac{2}{7}]_{Anna}\,\mathrm{role}(Brian) = Werewolf$$

Belief Quality



$$\mathcal{M} \models Q[\geq \tfrac{1}{7}]_{Anna}\, \mathrm{role}(Brian) = Rascal$$

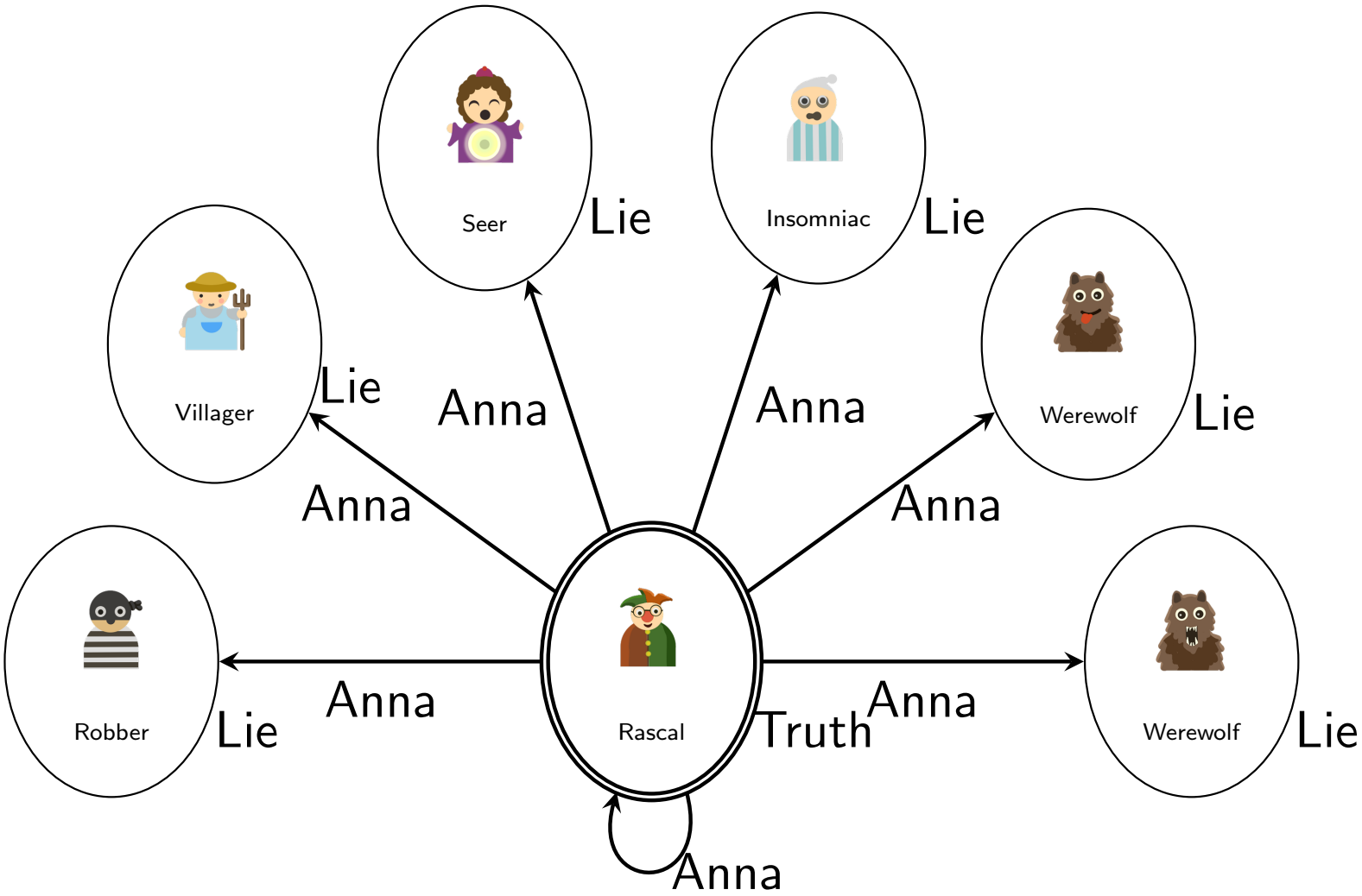
What about communication?

- Brian says (truthfully) that he is the Rascal
- What is Anna supposed to do with this information?
- Anna does not know that Brian is telling the truth

What about communication?

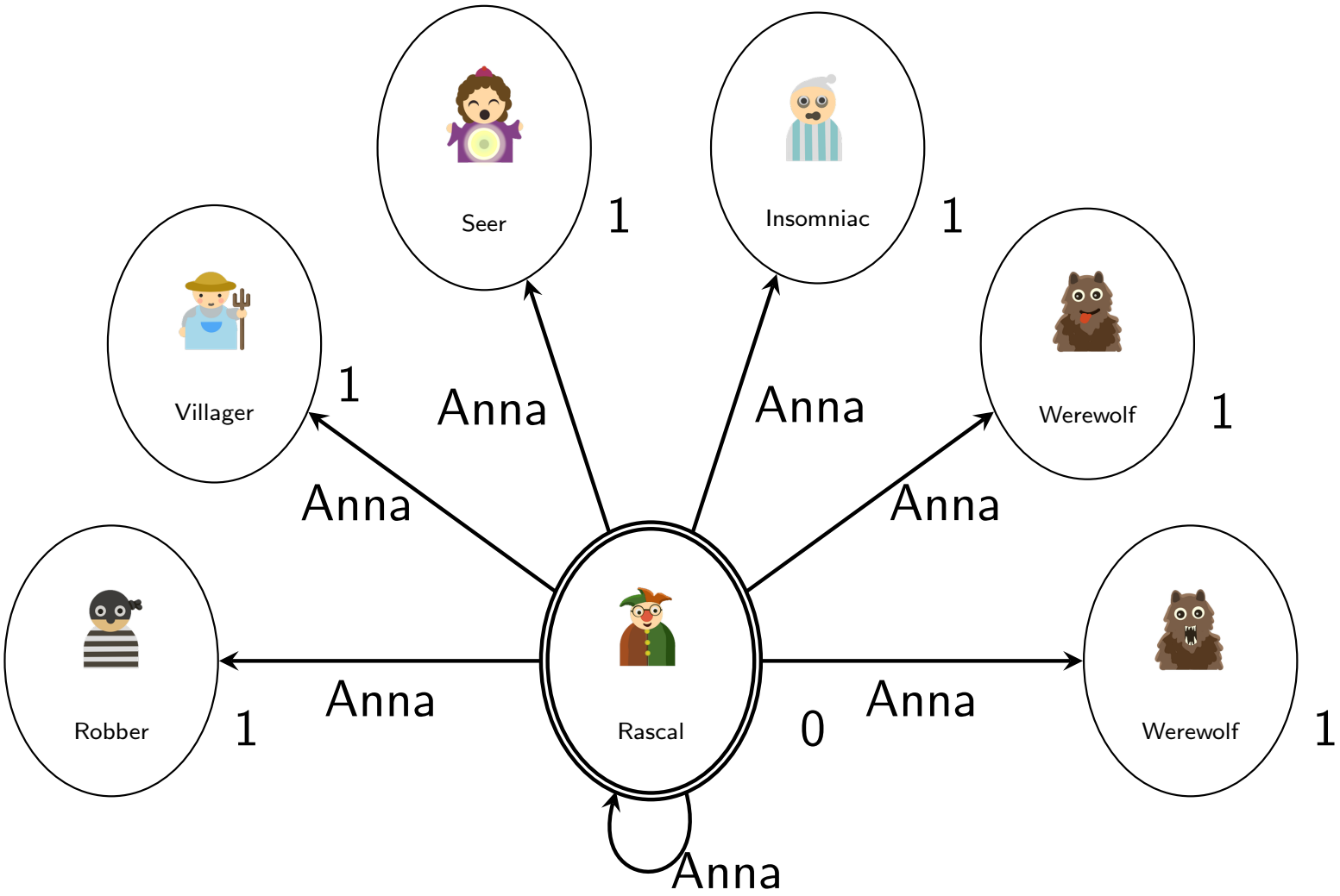
- Brian says (truthfully) that he is the Rascal
- What is Anna supposed to do with this information?
- Anna does not know that Brian is telling the truth
- But what if he is?

Belief Quality



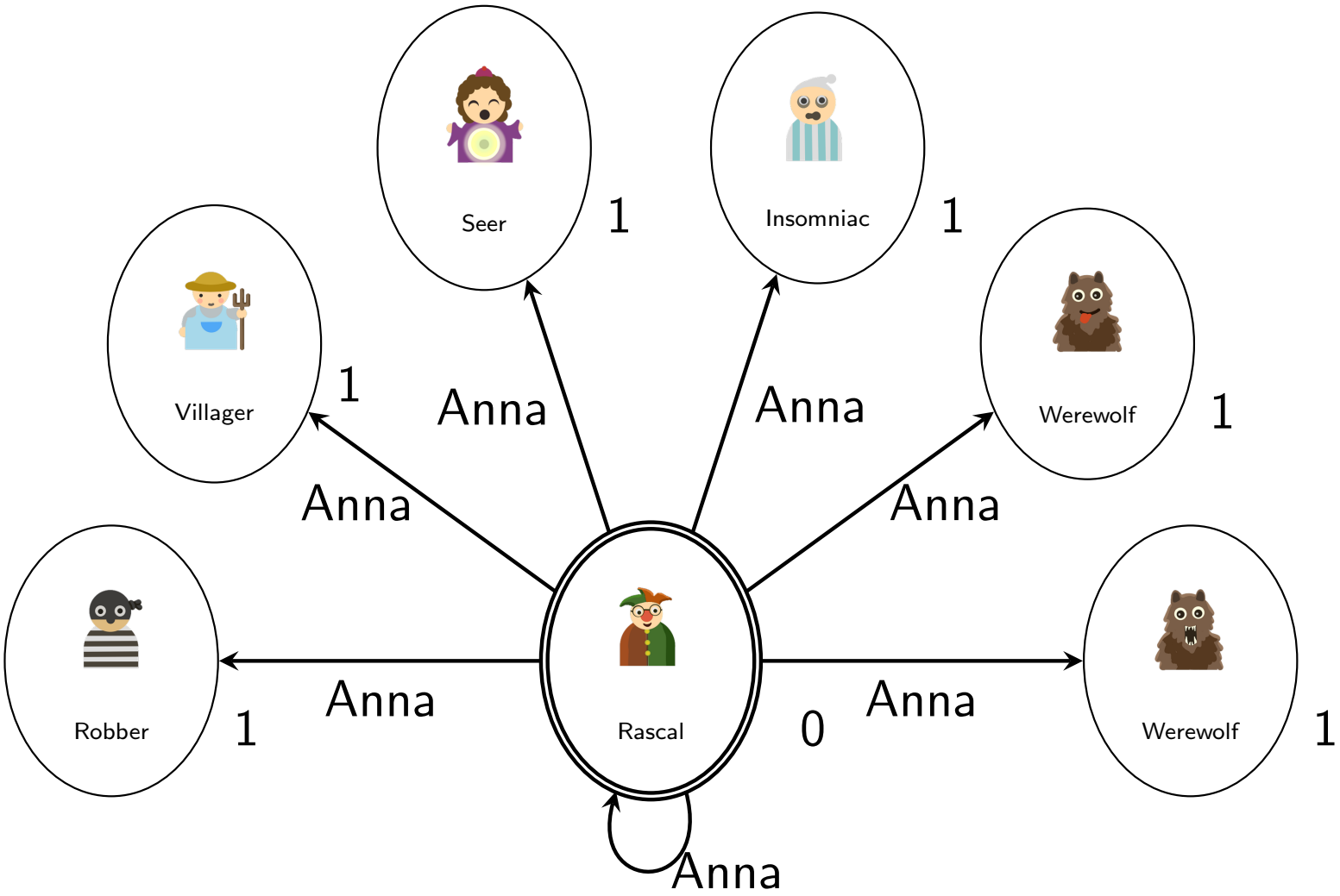
$\mathcal{M}$

Belief Quality



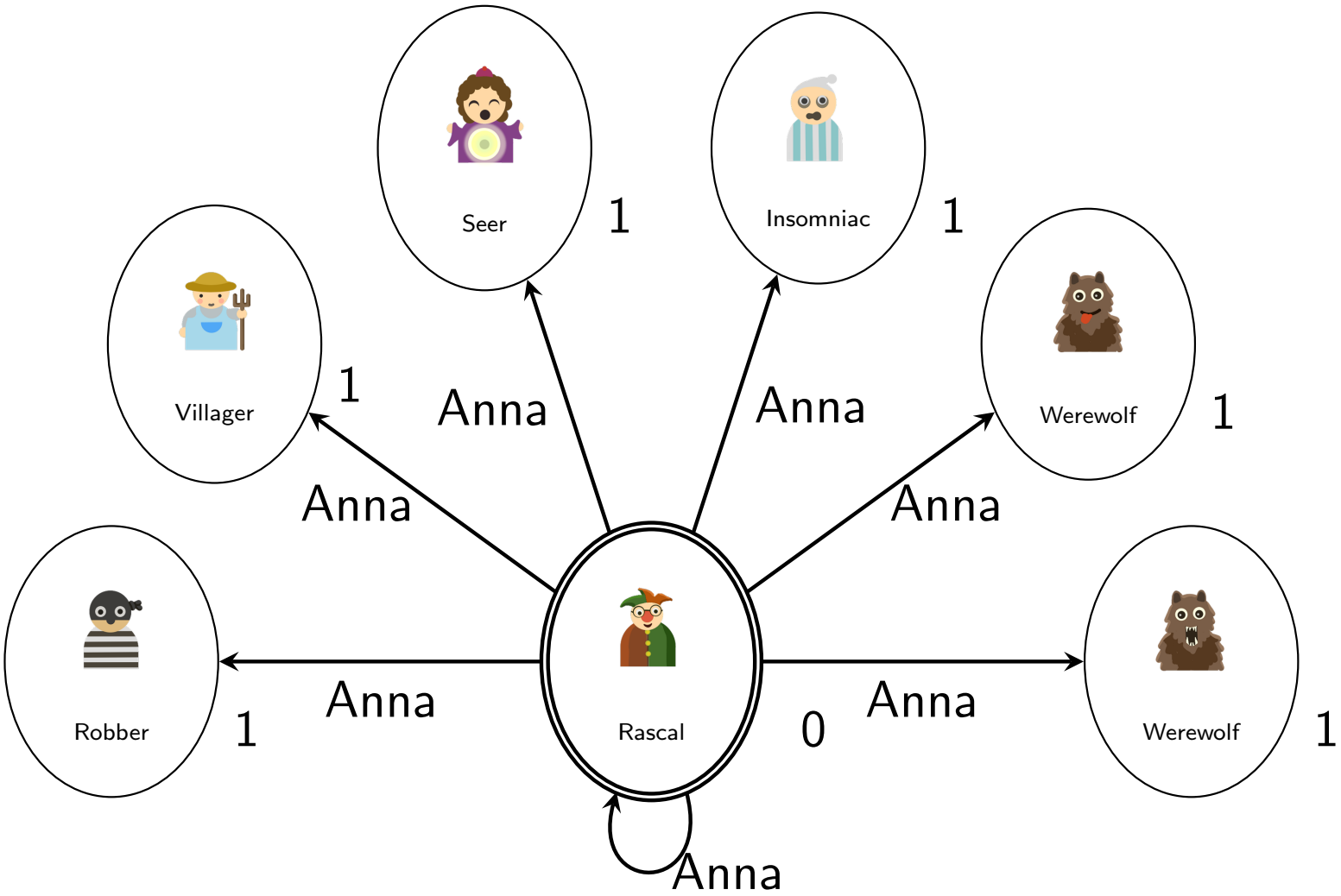
Higher numbers mean more lies, less likely to be true.

Belief Quality



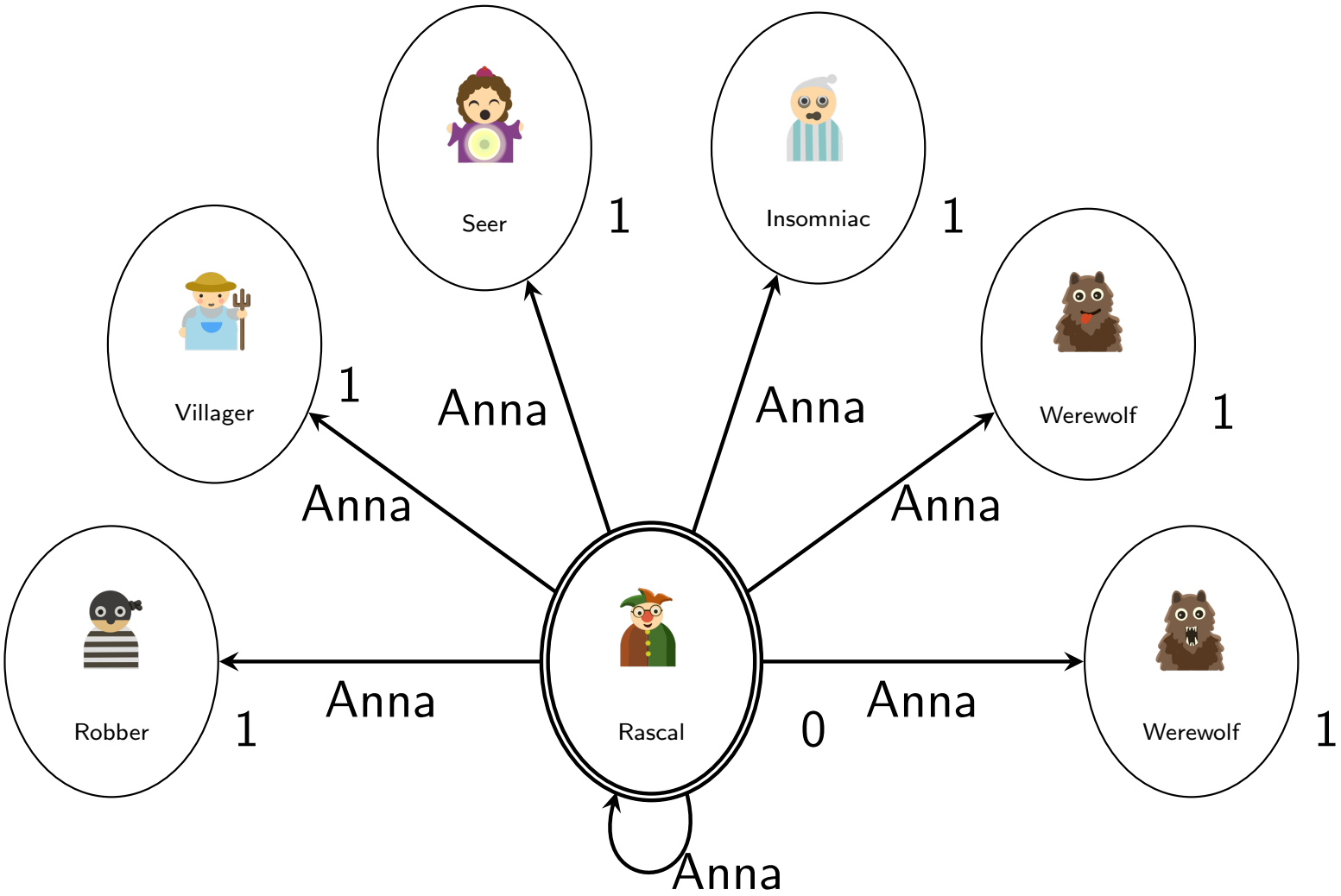
Instead of counting each world equally, use factor $\frac{1}{1+w}$.

Belief Quality



$$\mathcal{M} \models W[\geq \tfrac{1}{4}]_{Anna}\,\mathrm{role}(Brian) = Werewolf$$

Belief Quality



$$\mathcal{M} \models W[\geq \tfrac{1}{4}]_{Anna}\,\mathrm{role}(Brian) = Rascal$$

What does this have to do with planning?

- Brian says (truthfully) that he is the Rascal
- The outcome is *not* that Anna now believes him
- Anna thinks it is more likely that he is the Rascal, though

What does this have to do with planning?

- Brian says (truthfully) that he is the Rascal
- The outcome is *not* that Anna now believes him
- Anna thinks it is more likely that he is the Rascal, though
- Instead of planning to reach a goal: plan to maximize weighted quality

Encoding: Communicative actions

```
claimInitRole(p: Players, r: Roles, r1(p): Roles)
{
    precondition initrole(p) == r1;
    if (initrole(p) != r)
    {
        addWeight(1)
    }
    else
    {

    }
}
```

Goals

```
x = any(GoodRole)
    in
```

Goals

```
x = any(GoodRole)
    in

 W [>$cert] (self): role(self) == Werewolf:
```

Goals

```
x = any(GoodRole)
    in

 W [>$cert] (self): role(self) == Werewolf:

    Forall p in Players: W [>0.7] (p): (role(self) == x);
```

Application: One Night Ultimate Werewolf

Outline

Story generation example

- Detective story
- Moriarty kills the victim
- Goal: Moriarty believes that Sherlock believes that Watson is the murderer

Story generation example

```
kill(m(m): Suspects, v: Victims, g: Guns)
{
    precondition gunowner(g) == m;
    murderer(v) = m
}
```

Story generation example

```
take(m(m): Suspects, g: Guns)
{
    gunowner(g) = m
}
```

Story generation example

```
take(m(m): Suspects, g: Guns)
{
    gunowner(g) = m
}

put(m(m): Suspects, g: Guns, l: Locations)
{
    precondition gunowner(g) == m;
    gunowner(g) = Null;
    at(g) = l
}
```

Story generation example

```
take(m(m): Suspects, g: Guns)
{
    gunowner(g) = m
}

put(m(m): Suspects, g: Guns, l: Locations)
{
    precondition gunowner(g) == m;
    gunowner(g) = Null;
    at(g) = l
}

find(d: Detectives, g: Guns, l: Locations)
{
    precondition at(g) == l;
    suspect (d): gunowner(g) == owner(l)
}
```

Story generation example

```
Execute:
take(Moriarty,pistol)

goal: [] (Moriarty): [] (Sherlock):
        murderer(Victim) == Watson
```

Story generation example: Resulting Story

```
kill(Moriarty, Victim, pistol)

put(Moriarty, pistol, Shed)

Moriarty suspects find(Sherlock, pistol, Shed)
```

References

Thank you for your attention

- **Markus Eger** and Chris Martens. *Keeping the Story Straight: A Comparison of Commitment Strategies for a Social Deduction Game*. AIIDE 2018
- **Markus Eger** and Chris Martens. *Practical Specification of Belief Manipulation in Games*. AIIDE 2017
- Henry Mohr, **Markus Eger**, and Chris Martens. *Eliminating the Impossible: A Procedurally Generated Murder Mystery*. EXAG 2018
- **Markus Eger** and Chris Martens. *Character Beliefs in Story Generation*. INT 2017
- **Markus Eger** and Chris Martens. *Programming with Epistemic Logic*. OBT 2017