A Preliminary Approach on Modeling Machine Ethics

Maximilian Schlosser, Dominic Deckert

August 16, 2023

Philosophical Considerations

There are three main ethical schools:

- Duty ethics
  - duties provide base values
  - actions may not violate these values
- Consequentialism
  - the (positive) outcome of actions can be measured
  - actors should maximize the positive outcome
- Virtue ethics

We decided to use a combination of duty ethics and consequentialism.

Technical Considerations

We made the following technical considerations:

- there should be one base program
- programs should be generated using a well-defined method
- weak-completion semantics

The Context Operator

There are some cases where it is necessary to force the truth value of a variable to be either true or false. One example would be the default action rule, where the default action, in this case *watch*, should only be true if we know that no other action is taken.

To achieve this, we use the operator **ctxt:**

L	**ctxt(L)**
$\top$	$\top$
U	$\bot$
$\bot$	$\bot$

For example: $watch \leftarrow \neg\,\mathrm{ctxt}(switch), \neg\,\mathrm{ctxt}(push)$

Moral Decision Problems

- Moral Decision Problem: a set of modelling assumptions, abstracted from any concrete situation
- Problem instance: concrete situation to which the modelling assumptions apply

Idea: Description of problem instance $\longrightarrow$ logic program for problem instance

Trolley Problem

Modelling assumptions:

1. There is a segmentable track and a trolley racing down an initial track segment.
2. There are humans and objects on some track segments.
3. If the track branches, then there is a switch the agent can throw. There is an default setting for the switch.
4. With some tracks, the agent has the choice of moving humans/objects between them. This usually applies in only one direction.
5. If the trolley runs over any human/object, it gets killed/destroyed. This slows down the trolley.

Trolley Problem

Modelling assumptions:

1. There is a segmentable track and a trolley racing down an initial track segment.
2. There are humans and objects on some track segments.
3. If the track branches, then there is a switch the agent can throw. There is an default setting for the switch.
4. With some tracks, the agent has the choice of moving humans/objects between them. This usually applies in only one direction.
5. If the trolley runs over any human/object, it gets killed/destroyed. This slows down the trolley.

Trolley Problem

Modelling assumptions:

1. There is a segmentable track and a trolley racing down an initial track segment.
2. There are humans and objects on some track segments.
3. If the track branches, then there is a switch the agent can throw. There is an default setting for the switch.
4. With some tracks, the agent has the choice of moving humans/objects between them. This usually applies in only one direction.
5. If the trolley runs over any human/object, it gets killed/destroyed. This slows down the trolley.

Trolley Problem

Modelling assumptions:

1. There is a segmentable track and a trolley racing down an initial track segment.
2. There are humans and objects on some track segments.
3. If the track branches, then there is a switch the agent can throw. There is an default setting for the switch.
4. With some tracks, the agent has the choice of moving humans/objects between them. This usually applies in only one direction.
5. If the trolley runs over any human/object, it gets killed/destroyed. This slows down the trolley.

Trolley Problem

Modelling assumptions:

1. There is a segmentable track and a trolley racing down an initial track segment.
2. There are humans and objects on some track segments.
3. If the track branches, then there is a switch the agent can throw. There is an default setting for the switch.
4. With some tracks, the agent has the choice of moving humans/objects between them. This usually applies in only one direction.
5. If the trolley runs over any human/object, it gets killed/destroyed. This slows down the trolley.

Trolley Problem

Modelling assumptions:

1. There is a segmentable track and a trolley racing down an initial track segment.
2. There are humans and objects on some track segments.
3. If the track branches, then there is a switch the agent can throw. There is an default setting for the switch.
4. With some tracks, the agent has the choice of moving humans/objects between them. This usually applies in only one direction.
5. If the trolley runs over any human/object, it gets killed/destroyed. This slows down the trolley.

A problem instance is fully described by the orange components.

An Instance Description Language for the Trolley Problem

domain of track segments: $\mathbb{N}$ (for simplicity)

$Pred = \{\texttt{follows}^{(2)}, \texttt{object}^{(1)}, \texttt{human}^{(2)}, \texttt{switch}^{(3)}, \texttt{move}^{(2)}\}$

An *instance description D* is a finite, non-empty set of instantiated predicates that fulfills the consistency predicates.

Example: Footbridge Case
$D = \{\texttt{follows}(0,1), \texttt{follows}(1,2), \texttt{move}(3,1), \texttt{human}(2,5), \texttt{human}(3,1)\}$

An Instance Description Language for the Trolley Problem

domain of track segments: $\mathbb{N}$ (for simplicity)

$Pred = \{\texttt{follows}^{(2)}, \texttt{object}^{(1)}, \texttt{human}^{(2)}, \texttt{switch}^{(3)}, \texttt{move}^{(2)}\}$

An *instance description D* is a finite, non-empty set of instantiated predicates that fulfills the consistency predicates.

Example: Footbridge Case
$D = \{\texttt{follows}(0,1), \texttt{follows}(1,2), \texttt{move}(3,1), \texttt{human}(2,5), \texttt{human}(3,1)\}$

An Instance Description Language for the Trolley Problem

domain of track segments: $\mathbb{N}$ (for simplicity)

$Pred = \{\texttt{follows}^{(2)}, \texttt{object}^{(1)}, \texttt{human}^{(2)}, \texttt{switch}^{(3)}, \texttt{move}^{(2)}\}$

An *instance description D* is a finite, non-empty set of instantiated predicates that fulfills the consistency predicates.

Example: Footbridge Case
$D = \{\texttt{follows}(0,1), \texttt{follows}(1,2), \texttt{move}(3,1), \texttt{human}(2,5), \texttt{human}(3,1)\}$

An Instance Description Language for the Trolley Problem

domain of track segments: $\mathbb{N}$ (for simplicity)

$Pred = \{\texttt{follows}^{(2)}, \texttt{object}^{(1)}, \texttt{human}^{(2)}, \texttt{switch}^{(3)}, \texttt{move}^{(2)}\}$

An *instance description D* is a finite, non-empty set of instantiated predicates that fulfills the consistency predicates.

Example: Footbridge Case
$D = \{\texttt{follows}(0, 1), \texttt{follows}(1, 2), \texttt{move}(3, 1), \texttt{human}(2, 5), \texttt{human}(3, 1)\}$

Consistency Constraints

A *instance description D* is a finite, non-empty set of "facts" which fulfills the following consistency constraints:

- dense switch settings
- unique switch settings
- symmetricality of switches
- unique move origin
- unique move target
- unique human numbers
- symmetrical segment successors with `follows`
- at most two segment successors

Consistency Constraints

A *instance description D* is a finite, non-empty set of "facts" which fulfills the following consistency constraints:

- dense switch settings
- unique switch settings
- symmetricality of switches
- unique move origin
- unique move target
- unique human numbers
- symmetrical segment successors with `follows`
- at most two segment successors

Consistency Constraints

A *instance description D* is a finite, non-empty set of "facts" which fulfills the following consistency constraints:

- dense switch settings
- unique switch settings
- symmetricality of switches
- unique move origin
- unique move target
- unique human numbers
- symmetrical segment successors with `follows`
- at most two segment successors

Preconditions

Given: an *instance description D*
Calculate: the translated program *P*
New predicate symbols: $\texttt{segment}^{(1)}, \texttt{stop}^{(1)}, \texttt{obstacle}^{(1)}$ and abducibles for all actions (of the form $\texttt{switch\_(x, a)}, \texttt{move\_(x, y)}$)

Preconditions

Given: an *instance description D*
Calculate: the translated program *P*
New predicate symbols: `segment`$^{(1)}$, `stop`$^{(1)}$, `obstacle`$^{(1)}$ and abducibles for all actions (of the form `switch_(x, a)`, `move_(x, y)`)

World rules

- Observation rule:
```
on_track :- segment(0).
```
- Kill rule:
```
kill(X, Y) :- segment(X), human(X, Y), Y>0.
```
- Save rule:
```
save(X, Y) :- human(X, Y), -kill(X, Y).
```
- Stop rule:
```
stop(X) :- obstacle(X), segment(X).
```
- Obstacle rule:
```
obstacle(X) :- object(X).
obstacle(X) :- human(X, Y), Y>0.
```

World rules

- Observation rule:

```
on_track :- segment(0).
```

- Kill rule:

```
kill(X, Y) :- segment(X), human(X, Y), Y>0.
```

- Save rule:

```
save(X, Y) :- human(X, Y), -kill(X, Y).
```

- Stop rule:

```
stop(X) :- obstacle(X), segment(X).
```

- Obstacle rule:

```
obstacle(X) :- object(X).
obstacle(X) :- human(X, Y), Y>0.
```

World rules

- Observation rule:

```
on_track :- segment(0).
```

- Kill rule:

```
kill(X, Y) :- segment(X), human(X, Y), Y>0.
```

- Save rule:

```
save(X, Y) :- human(X, Y), -kill(X, Y).
```

- Stop rule:

```
stop(X) :- obstacle(X), segment(X).
```

- Obstacle rule:

```
obstacle(X) :- object(X).
obstacle(X) :- human(X, Y), Y>0.
```

Instance Rules - Switch

- Reach rule: if follows(x, y) $\in D$
  segment(y) :- segment(x), -ctxt(stop(x)).
- Switch rule: if switch (x, y, a) $\in D$, p is maximum switch setting
  segment(y) :- segment(x), -ctxt(stop(x)), switch_(x, a).}
- Switch consistency rule: if for all $0 < i \leq m$: switch (x, y_i, i) $\in D$, m is maximum switch setting
  false :- switch_(x, j), switch_(x, k).
  for all $0 < j < k \leq m$

Instance Rules - Switch

- Reach rule: <u>if</u> `follows(`x`, `y`)` $\in D$
  `segment(`y`) :- segment(`x`), -ctxt(stop(`x`)).`
- Switch rule: <u>if</u> `switch (`x`, `y`, `a`)` $\in D$, p is maximum switch setting
  `segment(`y`) :- segment(`x`), -ctxt(stop(`x`)), switch_(`x`, `a`).}`
- Switch consistency rule: <u>if</u> for all $0 < i \leq m$: `switch (`x`, `y_i`, `i`)` $\in D$, m is maximum switch setting
  `false :- switch_(`x`, `j`), switch_(`x`, `k`).`
  for all $0 < j < k \leq m$

Instance Rules - Shove

Intuition: If the agent moves humans, then the number on the target segment is the sum of humans on both segments, otherwise it is the original number

- Shove rule:

if move(z, x) $\in D$, $y = y\_1 + y\_2$
and for $i \in \{1, 2\}$: either human(x, y_i) $\in D$ or $y\_i = 0$
human(x, y) :- move_z_
human(z, 0) :- move_z. ,human(x, y_1) :- -ctxt(move_z).
human(z, y_2) :- -ctxt(move_z). }

Instance Rules - Shove

Intuition: If the agent moves humans, then the number on the target segment is the sum of humans on both segments, otherwise it is the original number

- Shove rule:

$\underline{\text{if}}$ $\texttt{move(}z\texttt{, }x\texttt{)} \in D$, $y = y\_1 + y\_2$
and for $i \in \{1, 2\}$: either $\texttt{human(x, }y\_\texttt{i)} \in D$ or $y\_\texttt{i} = 0$
$\texttt{human(}x\texttt{, }y\texttt{) :- move\_}z\texttt{\_}$
$\texttt{human(}z\texttt{, 0) :- move\_}z\texttt{.}$ $\texttt{,human(}x\texttt{, }y\_\texttt{1) :- -ctxt(move\_}z\texttt{).}$
$\texttt{human(}z\texttt{, }y\_\texttt{2) :- -ctxt(move\_}z\texttt{).}$ $\}$

Instance Rules - Move

- move rule : <u>if</u> $\texttt{object}(z), \texttt{move}(z,\ x) \in D$
  $\texttt{object}(x)\ \texttt{:-}\ \texttt{move\_}z\texttt{.}, \texttt{object}(z)\ \texttt{:-}\ \texttt{-ctxt(move\_}z\texttt{)}.$

End Rules

- End rule 1: <u>if</u> ab_1, ..., ab_p are all action abducibles in P_i
  $P_{i+1} = P_i \cup \{$`watch :- -ab_1, ... , -ab_p.,`$\}$
- End rule 2:
  <u>if</u> switch(x, y, 0) $\in D$
  and where exactly switch_(x, y_j abducibles in P_i for all $1 \leq j \leq p$
  segment(y) :- seg(x), -ctxt(stop(x)),
  -ctxt(switch_(x, y_1)), ..., -ctxt(switch_(x, y_p)).
- End rule 3: <u>if</u> x is neither move origin nor move target
  human(x, a).

End Rules

- End rule 1: <u>if</u> ab_1, ..., ab_p are all action abducibles in P_i
  $P_{i+1} = P_i \cup \{$`watch :- -ab_1, ... , -ab_p.,`$\}$
- End rule 2:
  <u>if</u> switch(x, y, 0) $\in D$
  and where exactly switch_(x, y_j abducibles in P_i for all $1 \leq j \leq p$
  segment(y) :- seg(x), -ctxt(stop(x)),
  -ctxt(switch_(x, y_1)), ..., -ctxt(switch_(x, y_p)).
- End rule 3: <u>if</u> x is neither move origin nor move target
  human(x, a).

End Rules

- End rule 1: <u>if</u> ab_1, ..., ab_p are all action abducibles in P_i
  $P_{i+1} = P_i \cup \{$`watch :- -ab_1, ... , -ab_p.,`$\}$
- End rule 2:
  <u>if</u> `switch(`x`, `y`, 0)` $\in D$
  and where exactly `switch_(`x`, `y_j abducibles in P_i for all $1 \leq j \leq p$
  `segment(`y`) :- seg(`x`), -ctxt(stop(`x`)),`
  `-ctxt(switch_(`x`, `y_1`)), ..., -ctxt(switch_(`x`, `y_p`)).`
- End rule 3: <u>if</u> x is neither move origin nor move target
  `human(`x`, `a`).`

Reasoning

To decide which action should be taken, we use a preference relation that is largely based on duty ethics:

1. Prefer actions that do not kill anyone.
2. Prefer actions that do not intentionally kill anyone.
3. If two actions both either kill or intentionally kill, prefer the action that saves more people.

The preference relation is defined on the least models of the weak completion (lmwc) of a program. To be able to compute the lmwc w.r.t. each possible course of action, we set the truth value of the abducibles using observations.

Preference Relation

The preference relation $\prec$ is defined as follows:

Let I_1, I_2 be Imcw of a given program with different choices. Let intKill and kill be predicates in the program that model intentional and unintentional kills respectively. Let save be the amount of people that are saved. I_1 is preferred over I_2 if:

$$I_1 \prec I_2 : \begin{cases} kill \notin I_1, kill \in I_2, intKill \notin I_1 \cup I_2 & (1) \\ kill \in I_1, kill \in I_2, intKill \notin I_1 \cup I_2, save_1 > save_2 & (2) \\ intKill \notin I_1, intKill \in I_2 & (3) \\ intKill \in I_1, intKill \in I_2, save_1 > save_2 & (4) \end{cases}$$

The Base Case

The base case of the trolley problem is rather simple:

- There is a trolley heading to a splitting track
- There are five humans on the right track
- There is one human on the left track

$D = \{\texttt{switch}(0,1,0), \texttt{switch}(0,2,1), \texttt{human}(1,5), \texttt{human}(2,1)\}$

The Base Case

The base case of the trolley problem is rather simple:

- There is a trolley heading to a splitting track
- There are five humans on the right track
- There is one human on the left track

$$D = \{\texttt{switch}(0{,}1{,}0), \texttt{switch}(0{,}2{,}1), \texttt{human}(1{,}5), \texttt{human}(2{,}1)\}$$

Step 1: World Rules

The world rules can be applied regardless of the contents of D. Therefore, we assume that they have already been applied:

$$P_0 = \begin{cases} \mathit{on\_track} & \leftarrow \mathit{seg}(0) \\ \mathit{kill}(X,Y) & \leftarrow \mathit{event}(X), \mathit{life\_danger}(X,Y), Y > 0 \\ \mathit{save}(X,Y) & \leftarrow \mathit{life\_danger}(X,Y), \neg \mathit{kill}(X,Y) \\ \mathit{stop}(X) & \leftarrow \mathit{obs}(X), \mathit{seg}(X) \\ \mathit{obs}(X) & \leftarrow \mathit{obj}(X) \\ \mathit{obs}(X) & \leftarrow \mathit{human}(X,Y), Y > 0 \end{cases}$$

$$D = \{\texttt{switch}(0,1,0),\texttt{switch}(0,2,1),\texttt{human}(1,5),\texttt{human}(2,1)\}$$

Step 2: Instance Rules

Next, apply the instance rules:

$$P_1 = P_0 \cup \{seg(2) \leftarrow seg(0), \neg ctxt(stop(0)), switch(0,1), \neg ctxt(switch(0,0))\}$$

Finally, apply the end rules:

$$P_2 = P_1 \cup \{watch \leftarrow \neg ctxt(switch(0,1)\}$$

$$P_3 = P_2 \cup \{human(1,5), human(2,1)\}$$

$$P_4 = P_3 \cup \{seg(1) \leftarrow seg(0), \neg ctxt(stop(0)), \neg ctxt(switch(0,1))\}$$

$D = \{\texttt{switch(0,1,0)}, \texttt{switch(0,2,1)}, \texttt{human(1,5)}, \texttt{human(2,1)}\}$

Step 2: Instance Rules

Next, apply the instance rules:

$$P_1 = P_0 \cup \{seg(2) \leftarrow seg(0), \neg ctxt(stop(0)), switch(0,1), \neg ctxt(switch(0,0))\}$$

Finally, apply the end rules:

$$P_2 = P_1 \cup \{watch \leftarrow \neg ctxt(switch(0,1)\}$$

$$P_3 = P_2 \cup \{human(1,5), human(2,1)\}$$

$$P_4 = P_3 \cup \{seg(1) \leftarrow seg(0), \neg ctxt(stop(0)), \neg ctxt(switch(0,1))\}$$

$D = \{\texttt{switch}(0{,}1{,}0){,}\texttt{switch}(0{,}2{,}1){,}\texttt{human}(1{,}5){,}\texttt{human}(2{,}1)\}$

Base Case

$$stop(X) \leftarrow obs(X), seg(X). \quad (1)$$
$$obs(X) \leftarrow human(X, Y). \quad (2)$$
$$obs(X) \leftarrow obj(X). \quad (3)$$
$$on\_track \leftarrow seg(0). \quad (4)$$
$$save(X, Y) \leftarrow human(X, Y), \neg kill(X). \quad (5)$$
$$kill(X, Y) \leftarrow human(X, Y), seg(X). \quad (6)$$
$$human(2, 1). \quad (7)$$
$$human(1, 5). \quad (8)$$

$$watch \leftarrow \neg\, \mathrm{ctxt}(switch(0, 1)). \quad (9)$$
$$seg(2) \leftarrow seg(0), \neg ctxt(stop(0)), switch(0, 1). \quad (10)$$
$$seg(1) \leftarrow seg(0), \neg ctxt(stop(0)), \neg ctxt(switch(0, 1)). \quad (11)$$

Minimal Explanations

$$lmwcs(P \cup (\{on\_track\}, \emptyset)) : \{watch, ontrack, seg(0), seg(1), stop(1), kill(1, 5)\}, \{save(1, 5)\}$$
$$lmwcs(P \cup (\{on\_track, switch(0, 1)\}, \emptyset)) : \{on\_track, seg(2), stop(2), kill(2, 1), save(1, 5)\}, \{seg(1), save(2, 1), kill(1, 5), watch\}$$

Generalizing the Trolley Problem

1. There is a segmentable track and a trolley racing down an initial track segment.
2. There are humans and objects on some track segments.
3. If the track branches, then there is a switch the agent can throw. There is an default setting for the switch.
4. With some tracks, the agent has the choice of moving humans/objects between them. This usually applies in only one direction.
5. If the trolley collides with any human/object, it gets killed/destroyed. This slows down the trolley.

Traits specific to trolley problems

Other Moral Decision Problems

- Notions of moral decision problem / problem instance apply to other problems as well: e.g. surgeon problem, ...
- Framework can be used especially well for temporal relations
- Extending the translation should be possible
- Renaming (for generality):

track segment	possible event
trolley position	current event
switch	choice of the agent

Other Moral Decision Problems

Notions of moral decision problem / problem instance apply to other problems as well: e.g. surgeon problem, ...
Framework can be used especially well for temporal relations
Extending the translation should be possible
Renaming (for generality):

track segment	possible event
trolley position	current event
switch	choice of the agent

Other Moral Decision Problems

- Notions of moral decision problem / problem instance apply to other problems as well: e.g. surgeon problem, ...
- Framework can be used especially well for temporal relations
- Extending the translation should be possible
- Renaming (for generality):

track segment	possible event
trolley position	current event
switch	choice of the agent

Problem

Modelling assumption specific to trolley problem: Stop after any collision
solution ideas:

- adding `stopping`$^{(1)}$ to the instance description predicates
  thus: making stopping an abnormal
- identifying certain consequences as stopping
- extending problem instance by a set of modelling assumptions