Towards a Computational Theory of Action, Causation and Power for Normative Reasoning

32nd Conference on Legal Knowledge and Information Systems

12 December 2019, Madrid

Giovanni Sileno (**g.sileno@uva.nl**),
Alexander Boer, Tom van Engers

Types of normative reasoning

- reasoning ***with*** norms: **applying norms** in the form of *directives* and *knowledge constructs* to interpret reality and decide what should be concluded or done.

Types of normative reasoning

- reasoning ***with*** norms: **applying norms** in the form of *directives* and *knowledge constructs* to interpret reality and decide what should be concluded or done.

This is a violation!

Types of normative reasoning

- reasoning ***about*** norms: reflecting on, evaluating, assessing, **deciding upon norms**

Types of normative reasoning

- reasoning ***about*** norms: reflecting on, evaluating, assessing, **deciding upon norms**

- ***internal view*:**
  - whether a norm is **valid** and **applicable** w.r.t. other norms

Types of normative reasoning

- reasoning ***about*** norms: reflecting on, evaluating, assessing, **deciding upon norms**

This is a violation!

- ***internal view*:**
  - whether a norm is **valid** and **applicable** w.r.t. other norms

Types of normative reasoning

- reasoning ***about*** norms: reflecting on, evaluating, assessing, **deciding upon norms**

REGULATORY SYSTEM

This is a violation!

is this directive valid and applicable?

- ***internal view*:**
  - whether a norm is **valid** and **applicable** w.r.t. other norms

Types of normative reasoning

- reasoning ***about*** norms: reflecting on, evaluating, assessing, **deciding upon norms**

- ***external views*:**
  - whether the norm is **effective** in guiding behaviour

Types of normative reasoning

- reasoning ***about*** norms: reflecting on, evaluating, assessing, **deciding upon norms**

REGULATORY SYSTEM

This is a violation!

- ***external views*:**
  - whether the norm is **effective** in guiding behaviour

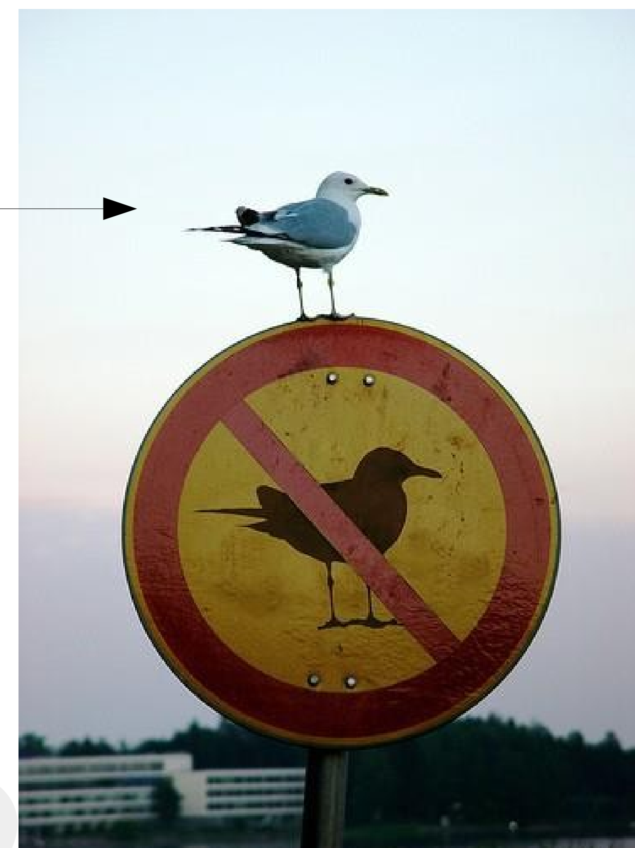
Types of normative reasoning

- reasoning ***about*** norms: reflecting on, evaluating, assessing, **deciding upon norms**

REGULATORY SYSTEM

IMPLEMENTATION

This is a violation!



- ***external views*:**
  - whether the norm is **effective** in guiding behaviour

Types of normative reasoning

- reasoning ***about*** norms: reflecting on, evaluating, assessing, **deciding upon norms**

by **WHOM?**

are violations monitored and settled?

is legal remedy settled after violation?

is legal remedy provided?

REGULATORY SYSTEM

IMPLEMENTATION

This is a violation!

- ***external views*:**
  - whether the norm is **effective** in guiding behaviour

Types of normative reasoning

- reasoning ***about*** norms: reflecting on, evaluating, assessing, **deciding upon norms**

by **WHOM? HOW?**

REGULATORY SYSTEM

are violations monitored and settled?

is legal remedy settled after violation?

IMPLEMENTATION

This is a violation!

is legal remedy provided?

is the implementation sustainable?

- ***external views*:**
  - whether the norm is **effective** in guiding behaviour
  - whether it is **efficient** in terms of costs

Types of normative reasoning

- reasoning ***about*** norms: reflecting on, evaluating, assessing, **deciding upon norms**

by **WHOM? HOW?**

are violations monitored and settled?

is legal remedy settled after violation?

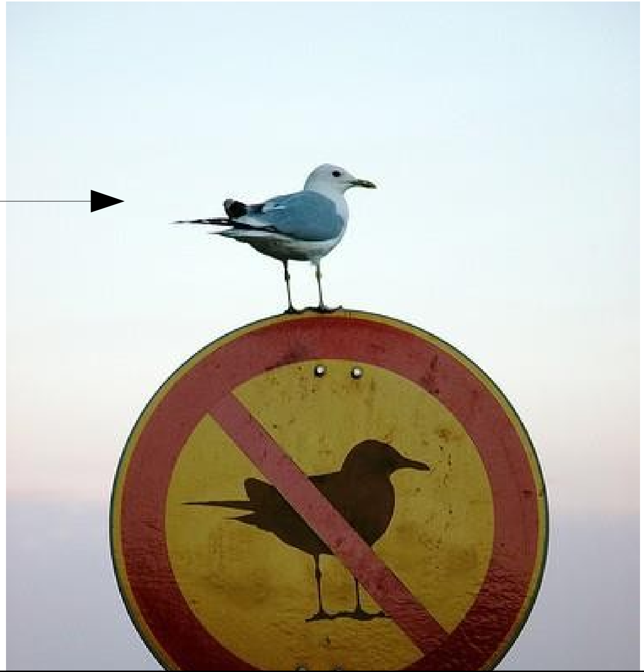
is legal remedy provided?

is the implementation sustainable?

REGULATORY SYSTEM

IMPLEMENTATION

This is a violation!



- ***external views*:**
  - whether the norm is **effective** in guiding behaviour

To effectively apply norms, we need a viable implementation!

Research context: Digital Market-Places (DMPs) infrastructures

legal norms	*rules of "society"*	
DMP policy	*rules of the "game"*	***these are about what ought to be (but may be violated)***
agreements, contracts	*ad-hoc rules set amongst "players"*	
transactions	*"rules" of the infrastructure*	***these are about what may be (possibility)***

Research context: Digital Market-Places (DMPs) infrastructures

legal norms	*rules of "society"*	
DMP policy	*rules of the "game"*	***these are about what ought to be (but may be violated)***
agreements, contracts	*ad-hoc rules set amongst "players"*	
transactions	*"rules" of the infrastructure*	***these are about what may be (possibility)***

operationalizing normative systems boils down to designing ***power structures distributed*** to **computational actors.**

Relevant concepts

- **ACTION:** *event* driven by an **AGENT**
- **CAUSATION:** mechanism producing consequences of *events*
- **POWER:** reification of **CAUSATION** associated to an **ACTION**

This paper presents a preliminary axiomatization based on *Logic Programming* constructs

Why Logic Programming?

- ***practical reasons***
  - tractability, scalability, programmability
  - “general” logic framework (no specific modal logics)

- ***strategic reasons***
  - general renewed interest towards LP
  - rule-based interpretations of ML black boxes

Action

Actions: levels of abstraction

- The same event can be described at different levels of abstraction.

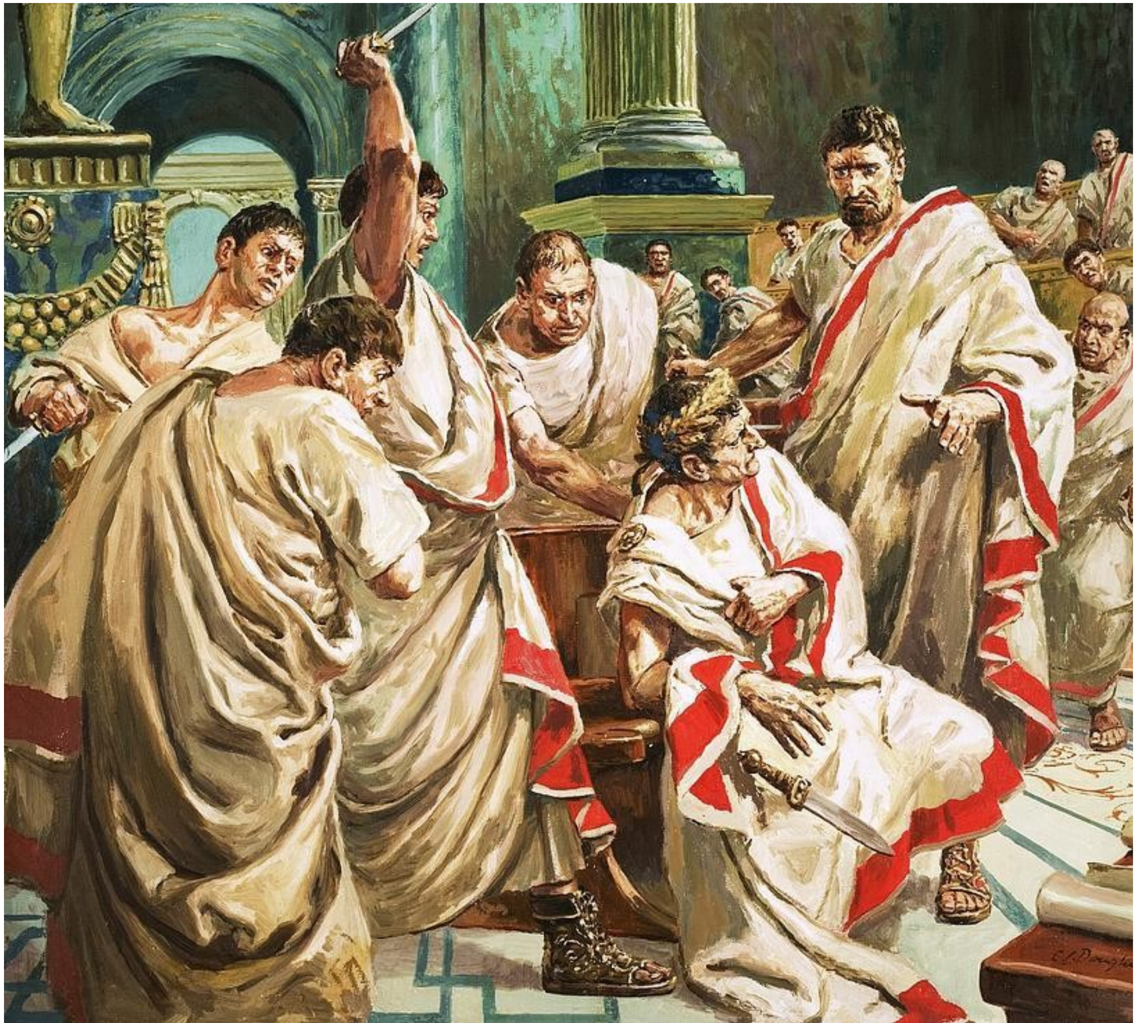
Brutus

stabbed task/operation

killed outcome

murdered intent

Caesar



Actions: characterizations

- By focusing on a certain action, we can recognize 3 characterizations:

procedural/Behavioural

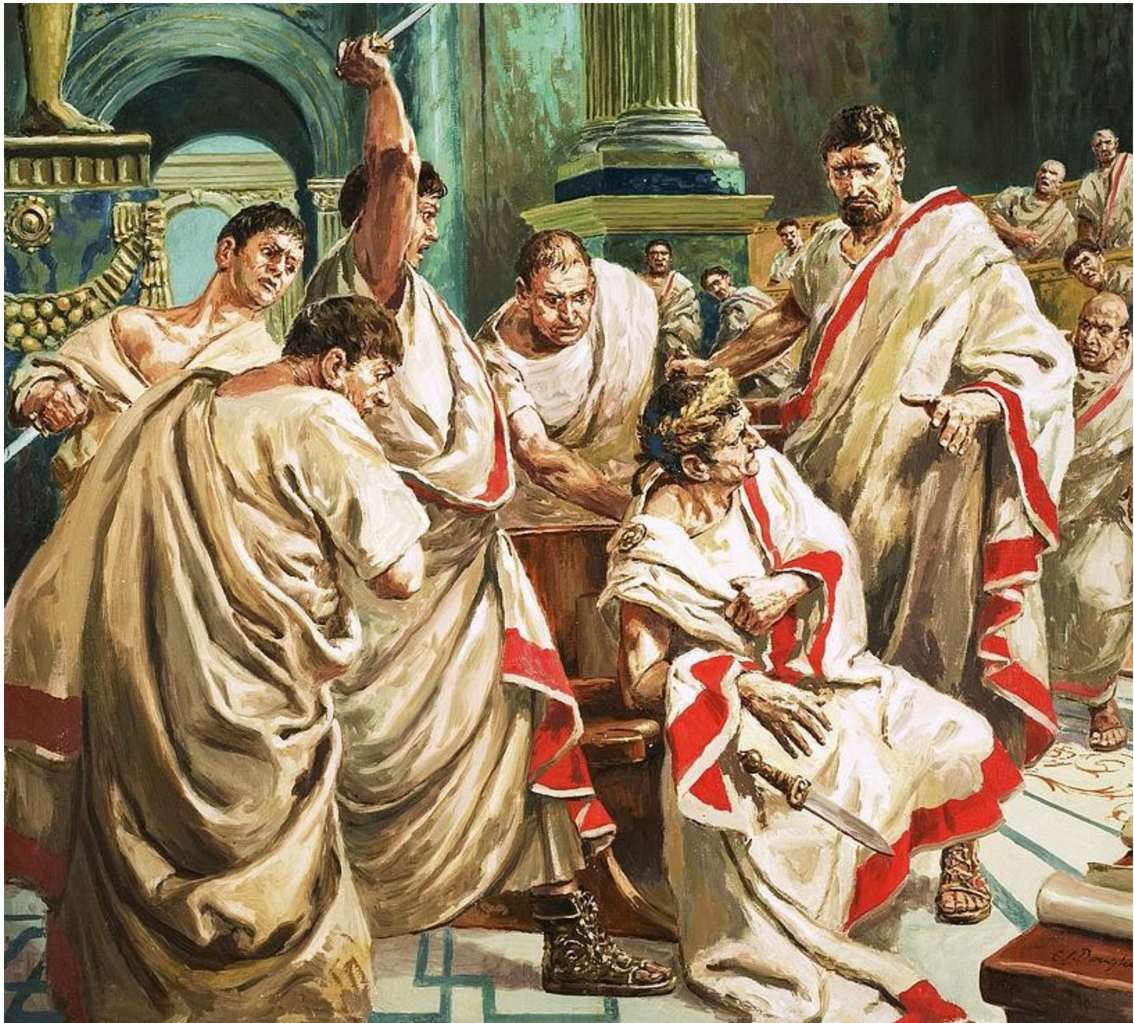
`performs(brutus, stabbing)`

productive

`brings(brutus, stabbed)`

intentional

`aims(brutus, stabbing)`



Definition of actions

- behavioural or procedural characterization

```
does(brutus, stabbing) <-> performs(brutus, stabbing).
```

- productive characterization (based on a *default rule*)

```
does(brutus, killing) <*> brings(brutus, dead).
```

- intentional or purposive characterization

```
does(brutus, murdering) <->
      aims(brutus, killing), does(brutus, killing).
```

Definition of actions

- behavioural or procedural characterization

```
does(brutus, stabbing) <-> performs(brutus, stabbing).
```

- productive characterization (based on a *default rule*)

```
does(brutus, killing) <*> brings(brutus, dead).
```

- intentional or purposive characterization

```
does(brutus, murdering) <->
      aims(brutus, killing), does(brutus, killing).
```

the paper presents several axioms linking the different characterizations...

“Default” mechanism <*>

- If an act has been completed, then performance has occurred:

```
brings(brutus, stabbed) -> performs(brutus, stabbing).
```

- performance is completed by default, unless it is known otherwise:

```
performs(brutus, stabbing), not neg(brings(brutus,
stabbed))      -> brings(brutus, stabbed).
```

“Default” mechanism <*>

- If an act has been completed, then performance has occurred:

```
brings(brutus, stabbed) -> performs(brutus, stabbing).
```

- performance is completed by default, unless it is known otherwise:

```
performs(brutus, stabbing), not neg(brings(brutus,
stabbed))      -> brings(brutus, stabbed).
```

strong negation

default negation

Perfect/imperfect actions

- Let us consider actions identified by a task description **`A`** and an outcome description **`R`**, related by the predicate **`actionResult/2`**
- The following qualifications of an action A can be defined as **`does(X, A), actionResult(A, R)`** and these other conditions:
  - *perfect action*: **`brings(X, R)`**
  - *imperfect action:* **`neg(brings(X, R))`**
  - *ongoing action:* **`not(brings(X, R))`**
  - *successful intention:* **`aims(X, R), brings(X, R)`**
  - *failed intention:* **`aims(X, R), neg(brings(X, R))`**
  - *ongoing attempt:* **`aims(X, A), not(brings(X, R))`**

Negated actions

- Actions can be then defined ***negatively***, or better, in terms of
  - **failure,** by relying on the idea of imperfection:

    ```
    does(X, neg(A)) <-> imperfect(does(X, A)).
    ```

  - **omission,** as not initiated execution:

    ```
    neg(does(X, A)).
    ```

Causation

Causation

- Reactive rules, represented e.g. in the form of a ***event-condition-action*** **(ECA)** rule, provide a primitive computational construct reifying **symbolic causation**:

```
performs(X, A) : initiates(A, R) => +R.      % initiation of r
performs(X, A) : terminates(A, R) => -R.   % termination of r
```

Causation

- Reactive rules, represented e.g. in the form of a ***event-condition-action*** **(ECA)** rule, provide a primitive computational construct reifying **symbolic causation**:

```
performs(X, A) : initiates(A, R) => +R.     % initiation of r
performs(X, A) : terminates(A, R) => -R.  % termination of r
```

- *Why ECA rules? What if we make explicit the temporal annotation and express causation as logical dependency?*

```
performs(X, A, T), initiates(A, R), neg(holds(R, T-1)) ->
holds(R,T).
performs(X, A, T), terminates(A, R), holds(R, T-1)) ->
neg(holds(R,T)).
```

Causation

- Reactive rules, represented e.g. in the form of a ***event-condition-action*** **(ECA)** rule, provide a primitive computational construct reifying **symbolic causation**:

```
performs(X, A) : initiates(A, R) => +R.      % initiation of r
performs(X, A) : terminates(A, R) => -R.   % termination of r
```

- *Why ECA rules? What if we make explicit the temporal annotation and express causation as logical dependency?*

```
performs(X, A, T), initiates(A, R), neg(holds(R, T-1)) ->
holds(R,T).
performs(X, A, T), terminates(A, R), holds(R, T-1)) ->
neg(holds(R,T)).
```

...wrong! Missing inertia and other properties, etc. we need to refer to ***Event Calculus*** or similar machinery!

Power

Modeling power

- Power—of an agent **X** towards an object **Y** to obtain a consequence **R** (concerning **Y**) by performing an action **A**—can be seen as the reification of a causal mechanism:

```
power(X, Y, A, R) <-> [performs(X, A) => +R(Y)].
```

Modeling power

- Power—of an agent **X** towards an object **Y** to obtain a consequence **R** (concerning **Y**) by performing an action **A**—can be seen as the reification of a causal mechanism:

```
power(X, Y, A, R) <-> [performs(X, A) => +R(Y)].
```

- The biconditional can be nested in the reactive rule...

```
performs(X, A) : power(X, Y, A, R) => +R(Y).
```

Modeling power

- Power—of an agent `X` towards an object `Y` to obtain a consequence `R` (concerning `Y`) by performing an action `A`—can be seen as the reification of a causal mechanism:

```
power(X, Y, A, R) <-> [performs(X, A) => +R(Y)].
```

- The biconditional can be nested in the reactive rule...

```
performs(X, A) : power(X, Y, A, R) => +R(Y).
```

the `initiates/2` predicate seen above is nothing else than a coarser description of `power/4 !!`

Modeling power

- Power—of an agent **`X`** towards an object **`Y`** to obtain a consequence **`R`** (concerning **`Y`**) by performing an action **`A`**—can be seen as the reification of a causal mechanism:

```
power(X, Y, A, R) <-> [performs(X, A) => +R(Y)].
```

- The biconditional can be nested in the reactive rule...

```
performs(X, A) : power(X, Y, A, R) => +R(Y).
```

the **`initiates/2`** predicate seen above is nothing else than a coarser description of **`power/4 !!`**

- The paper elaborates on related concepts as ability, susceptibility, negative power, etc.

Example of application: Interference

- Problem: set *protection measures against interference* as for ***freedom of speech***. But what is interference?

Example of application: Interference

- Problem: set *protection measures against interference* as for ***freedom of speech***. But what is interference?
- An action **`IA`** interferes with an action **`A`** if, when the first is performed, it inhibits the outcome usually expected for performing the second.

Example of application: Interference

- Problem: set *protection measures against interference* as for ***freedom of speech***. But what is interference?
- An action **`IA`** interferes with an action **`A`** if, when the first is performed, it inhibits the outcome usually expected for performing the second.
- Interference can be expressed in terms of power!

Structural interference

- Problem: set *protection measures against interference* as for ***freedom of speech***. But what is interference?
- An action **`IA`** interferes with an action **`A`** if, when the first is performed, it inhibits the outcome usually expected for performing the second.
- Interference can be expressed in terms of power!

```
% structural interference (disabling, specified at event
level)

power(Z, power(X, Y, A, R), IA, neg)
 <-> [ performs(Z, IA) => +neg(power(X, Y, A, R)). ]
```

Contingent interference

- Problem: set *protection measures against interference* as for ***freedom of speech***. But what is interference?
- An action **`IA`** interferes with an action **`A`** if, when the first is performed, it inhibits the outcome usually expected for performing the second.
- Interference can be expressed in terms of power!

```
% contingent interference (at object level, neglecting T)

power(Z, power(X, Y, A, R), IA, neg)
   <-> [ not performs(Z, IA) -> power(X, Y, A, R).
             performs(Z, IA) -> neg(power(X, Y, A, R)). ]
```

Conclusions

- For operationalization, normative systems need to seen as **social infrastructures**.

Conclusions

- For operationalization, normative systems need to seen as **social infrastructures**.

→ ***design of power structures is a crucial step!***

Conclusions

- For operationalization, normative systems need to seen as **social infrastructures**.

➡ ***design of power structures is a crucial step!***

- The paper presents our starting point for an axiomatization of power structures in a LP setting. Future work will refine and extend it to a wider number of institutional patterns (*ex-ante vs ex-post*, punishment vs reward-based enforcement, delegation, etc.) and concepts (recklessness, negligence, etc.).