Utilizing Permission Norms in BDI Practical Normative Reasoning

Wagdi Alrawagfeh

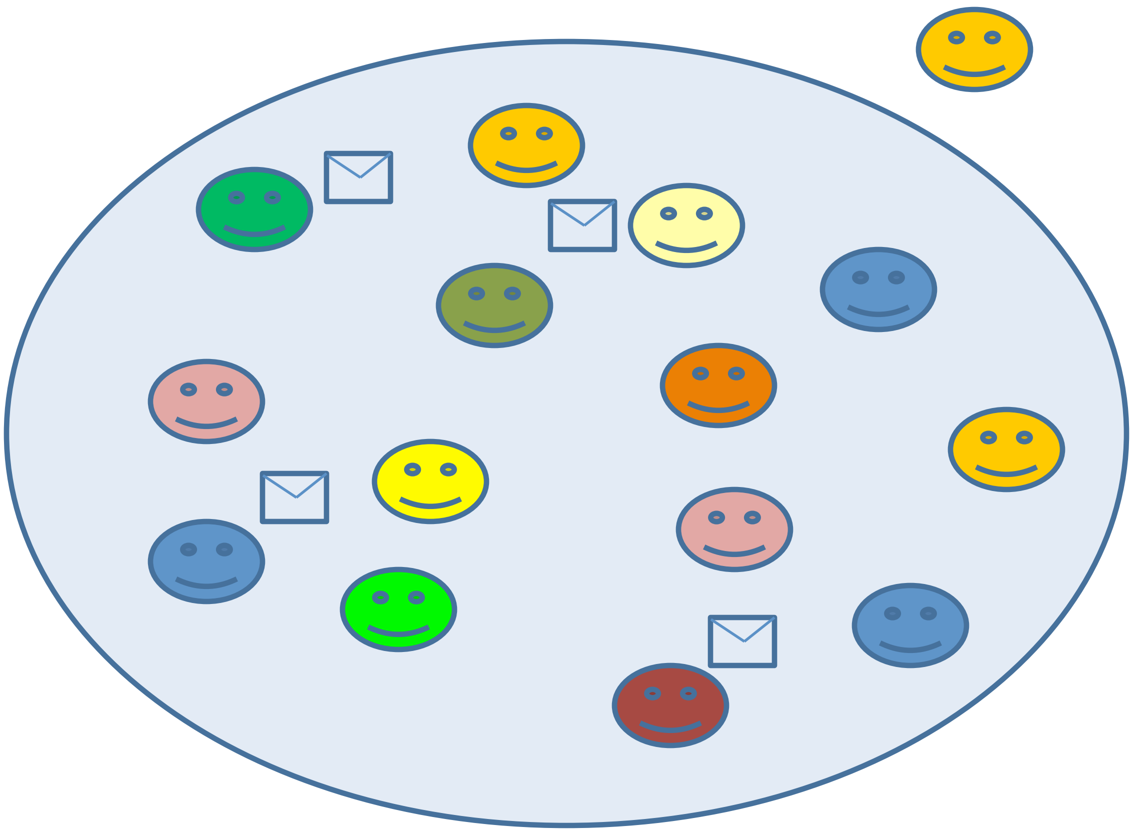
Computer Science Department
Memorial University of Newfoundland
St.John's, NL, Canada

Felipe Meneguzzi

School of Computer Science
Pontifical Catholic University of Rio Grande do Sul
Porto Alegre, RS, Brazil

Introduction

•Open Multiagent Systems



Heterogeneous

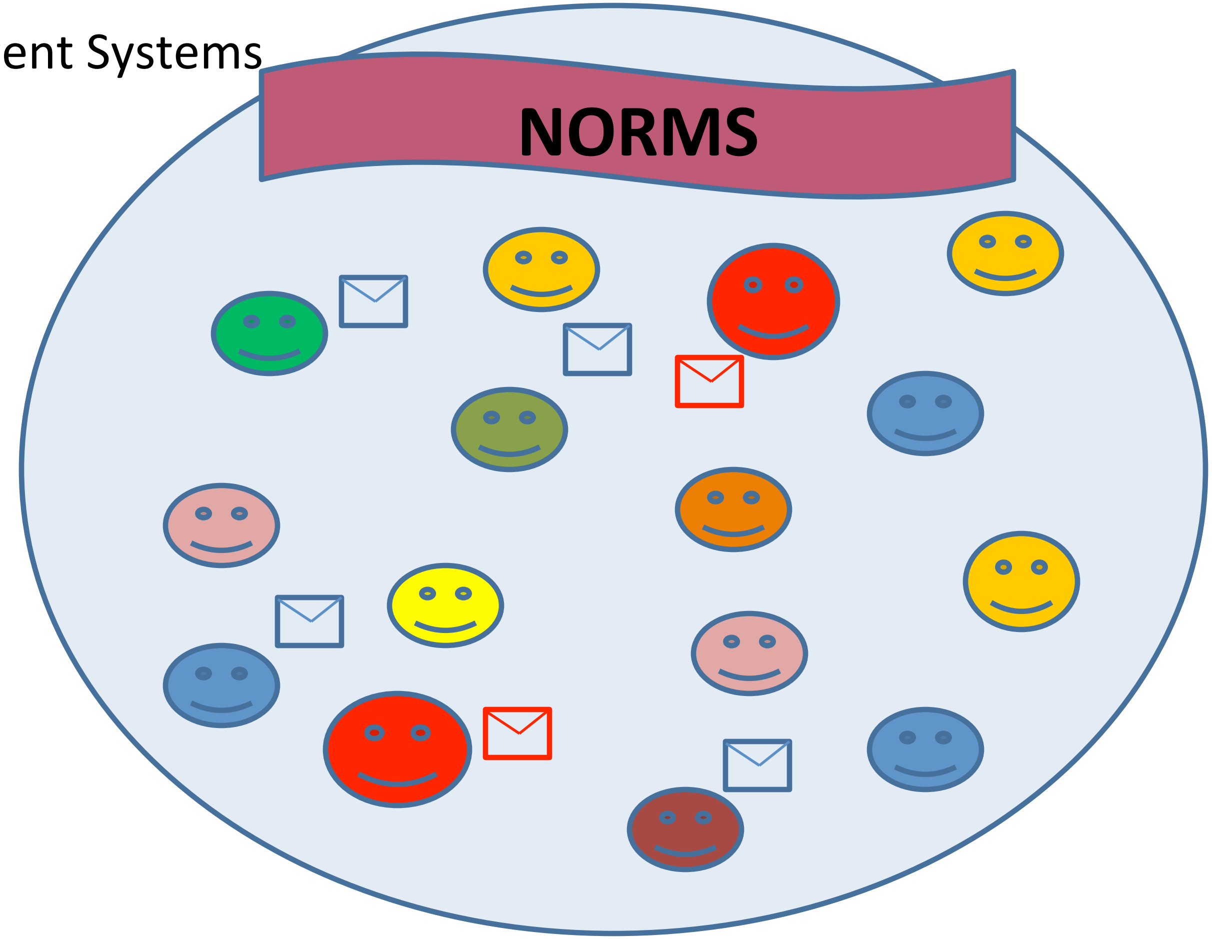
Autonomous

Self-Interested

Leave and join

Introduction

- Multiagent Systems



Dynamic

Heterogeneous

Autonomous

Self-Interested

Leave and join

Motivation

- Two major “choices” w.r.t. concrete implementations of normative agents
  - Regimentation (norms cannot be violated, i.e. hard constraints)
  - Enforcement (violations can occur, i.e. soft constraints)
- Enforcement makes runtime agent reasoning more complex

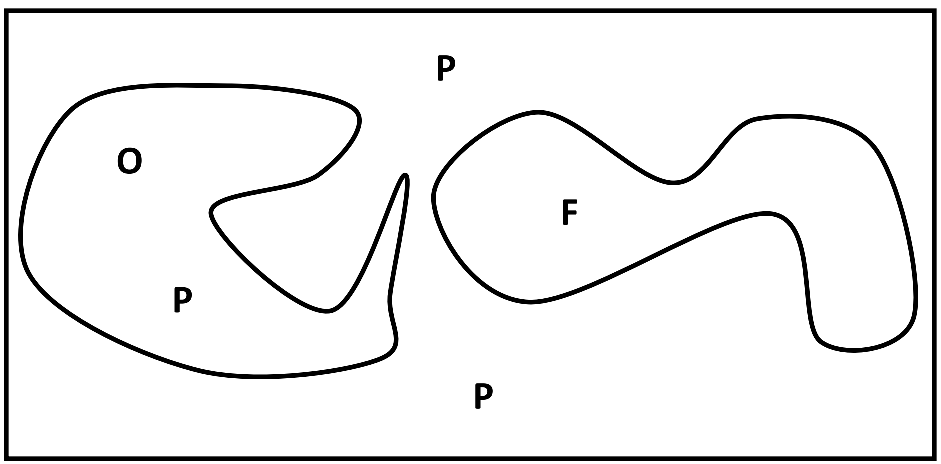
Motivation

- Much work on agent reasoning assume norms are:
  - Known in advance or agents are always aware of all norms
  - Ultimately decomposable into prohibitions/ obligations
  - Thus, no room for uncertainty about normative state
- We use reasoning about permissions to account for uncertainty
  - Relation to work on norm identification (assumed to exist)

Motivation

- Sealing principle

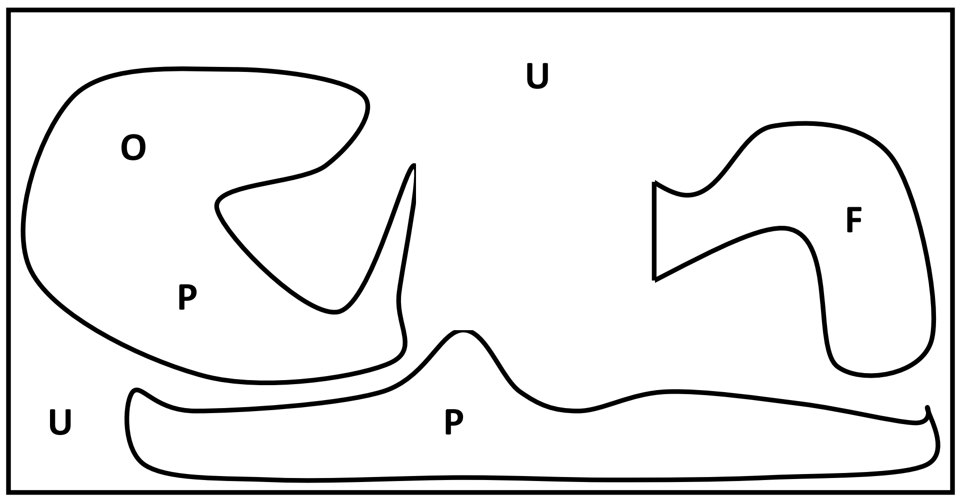
"whatever is not prohibited is permitted"



Complete knowledge of agent
about normative states

Motivation

- Incomplete knowledge → permission norm is significant



Background – Event Calculus

- Event Calculus
  - Framework in logic programming to represent and reason about actions and their consequences
  - Simple and widely used
    (usually implemented in Prolog)
- Jason
  - BDI-based programming language

Event Calculus

Predicate	Meaning
happens(A,T)	Action A occurs at time T
holdsAt(F,T)	Fluent F is true at time T
terminate(A,F,T)	Occurrence of action A at time T will make fluent F false after time T
initiates(A,F,T)	Occurrence of action A at time T will make fluent F true after time T
clipped(T,F,Tn)	Fluent F is terminated between time T and Tn
<, >, <=, >=	Standard order relation for time

Event Calculus

Predicate	Meaning
between(A,T1,T2)	Action A occurred after time T1 and before T2
initiatesAt(A,F,T1,T2)	The occurrence of action A at T1 will make fluent F true after T2, when T1≤ T2.
terminatesAt(A,F,T1,T2)	The occurrence of action A at time T1 makes fluent F false at time T2

Background - Jason

- BDI-based programming language
- Implementation of an extended version of the AgentSpeak(L) formalism/APL

```
triggering_event : context
     <- body.
```

- Supports a subset of logic programming constructs from Prolog

Norm Representation

- We define a norm as a tuple

  N = <D, C, Seq, S, R>

  D: *F, O* or *P*

  C: Context

  β: world state, defined via predicate holdsAt

  α: sequence of actions, defined via EC formula

  Seq: sequence of action(s) that agents are forbidden to perform or obliged to perform

  S: sanction

  R: reward

Norm Representation

- Permission norm representation:
  initiatesAt(An,pRew(Nid),Tn,Tn+1) :-
  C, happens(A1,T1) &…
  & happens(An,Tn) &
  T1<T2< & …& <Tn .
- Prohibition and Obligation norms represented in a similar way.

BDI agent normative reasoning

- In order to reason about plans in BDI agents, we define two key fluents
  - **help(Plan)** fluent will be true if executing Plan ends up with more rewards than punishments. That based on F and O norms.
  - **safe(Plan)** fluent will be true if executing Plan ends conforms with more permission norms

BDI agent normative reasoning

- Axioms:

EC1: *clipped (T1, F, T4) :- happens(A, T2) & terminatesAt(A, F, T2,T3) & T1< T2 & T2≤T3 & T3<T4*

EC3’: *holdsAt (F, T3) :- happens(A, T1) & initiatesAt(A, F, T1,T2) & T1≤ T2 & T2<T3 & not clipped (T2, F, T3)*

Ax1: between(A,T1,T2) :- happens(A, T) & T1<T & T<T2

Ax2: terminatesAt(*,help(P),T1,T2):- happens(*,T1)

Ax3: terminatesAt(*,safe(P),T1,T2):- happens(*,T1)

Ax4: terminatesAt(*,fPun(I,S),T1,T2):- happens(*,T1)

Ax5: terminatesAt(*,oPun(I,S),T1,T2):- happens(*,T1)

Ax6: terminatesAt(*,oRew(I,R),T1,T2):- happens(*,T1)

Ax7: terminatesAt(*,pRew(I,R),T1,T2):- happens(*,T1)

BDI agent normative reasoning

- **helpful-rule:**

initiatesAt(A,help(Plan_i),T1,T2):-
.findall(V1,holdsAt(oRew(_,V1),T2+1), Wins)
& .findall(V2,holdsAt(fPun(_,V2),T2+1), Loses1)
& .findall(V3,holdsAt(oPun(_,V3),T2+1), Loses2)
& goalpreference(G,Points) & Points + sum(Wins)-sum(Loses1) - sum(Loses2)> 0.

- **safe-rule:**

initiatesAt(A,safe(Plan_i),T1,T2):-
.findall(V1,holdsAt(pRew(_,V1),T2+1),Count).

Experiments

- Environment (based on Gold miners in Jason):
  - Grid-like territory with gold and silver pieces scattered
  - Three agents; best-agent and best-safest-agent and monitor-agent

Experiments

- Goal : collect gold to the depot.
- Agents have same plans for achieving the goal:

  1- collect gold to the silver depot

  2- collect gold to the gold depot.

  3- collect gold to the gold depot and silver to gold depot.

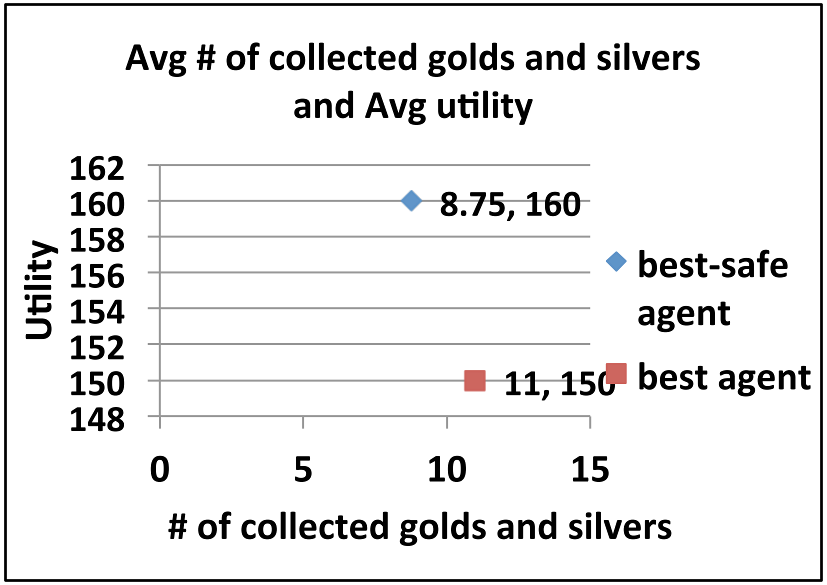
  4- collect gold to the gold depot and collect another gold to the gold depot.

  5- collect gold to the gold depot and collect silver to silver depot .

Experiments - Norms

- All agents are aware of at least the following norms:
  - It is prohibited to drop gold in the silver depot if the gold depot is not full, the sanction value is 5
  - It is prohibited to carry more than one gold piece at the same time, the sanction value is 10
  - It is obligatory to collect silver immediately after collecting gold: the sanction value is 10, the reward for compliance is 10.
- The best-safest-agent is also aware that:
  - It is permitted to drop gold in gold depot
  - It is permitted to drop silver in silver depot
- Finally, the monitor agent knows, and enforces an unknown norm:
  - It is prohibited to drop silver in the gold depot if the silver depot is not full, the sanction value is 10.

Experiments



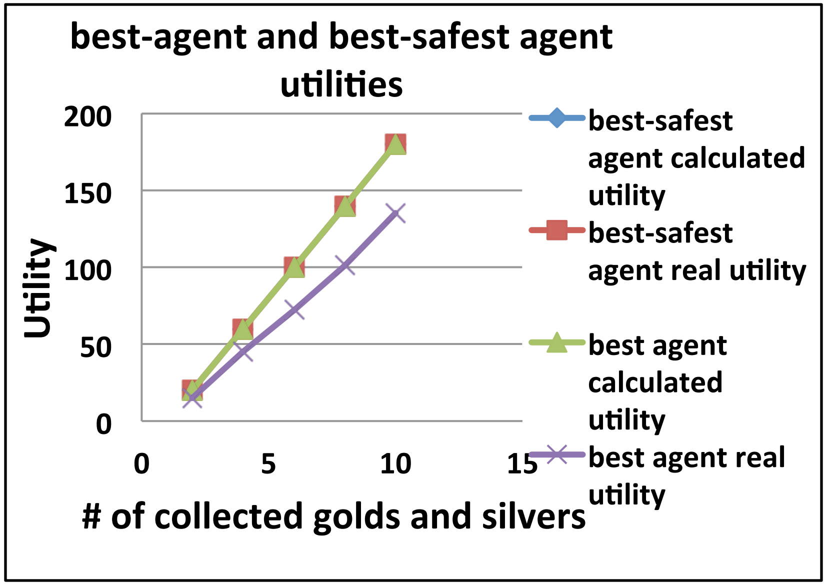


Fig.3 Shows the average collected gold and silver pieces and the ultimate achievement utilities for *best-agent and best-safest-agent.*

Fig.4 Shows the calculated/ predicted utility for the *best-safest-agent and best-agent.*

Conclusion

- We developed a norm reasoning mechanism that uses permissions to account for uncertainty
- Using permission norms gives agents the ability to have preference over plans
  - plans containing actions that are known to be permitted are preferable over plans that contain actions whose normative status is unknown

Future work

- Perform further experiments to study the time efficiency of our practical normative reasoning mechanism.
- Compare our *best-safest-agent with other BDI norm aware agents in the literature*