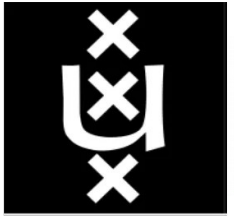


Integrating CP-Nets in Reactive BDI Agents

Mostafa Mohajeri (m.mohajeriparizi@uva.nl),
Giovanni Sileno, Tom van Engers
University of Amsterdam

30th October 2019, PRIMA2019 @ Torino University

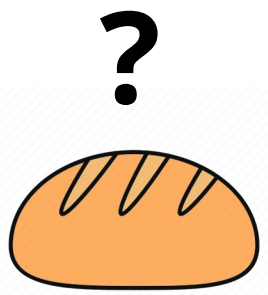
Outline

- Motivation
- BDI agents’ procedural knowledge
- Explicit preferences language: CP-Nets
- Reactivity as plan selection without reflection

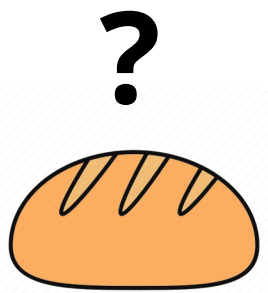
Outline

- Motivation
- BDI agents’ procedural knowledge
- Explicit preferences language: CP-Nets
- Reactivity as plan selection without reflection

Preferences affect choices



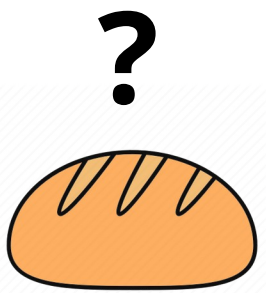
Preferences can be verbalized



Brown Bread > White Bread

Fresh > Stale

Preferences can be verbalized



Do we *reflect* on them every time?

Brown Bread > White Bread

Fresh > Stale

Motivation

- BDI (*belief-desire-intention*) agents are a powerful ABM (*agent-based modeling*) tool
- Main BDI frameworks lack explicit preferences
  - ex. Jason (AgentSpeak), 2APL/3APL, GOAL, Jadex
- Questions:
  - *How to represent preferences?*
  - *How to act based on preferences?*
- Goal: **integrate preferences while keeping reactivity**

Outline

- Motivation
- BDI agents’ procedural knowledge
- Explicit preferences language: CP-Nets
- Reactivity as plan selection without reflection

BDI procedural knowledge

- Plans as **Goal-Plan rules**: *means for achieving a goal*
- Formulated as $<e,c,p>$

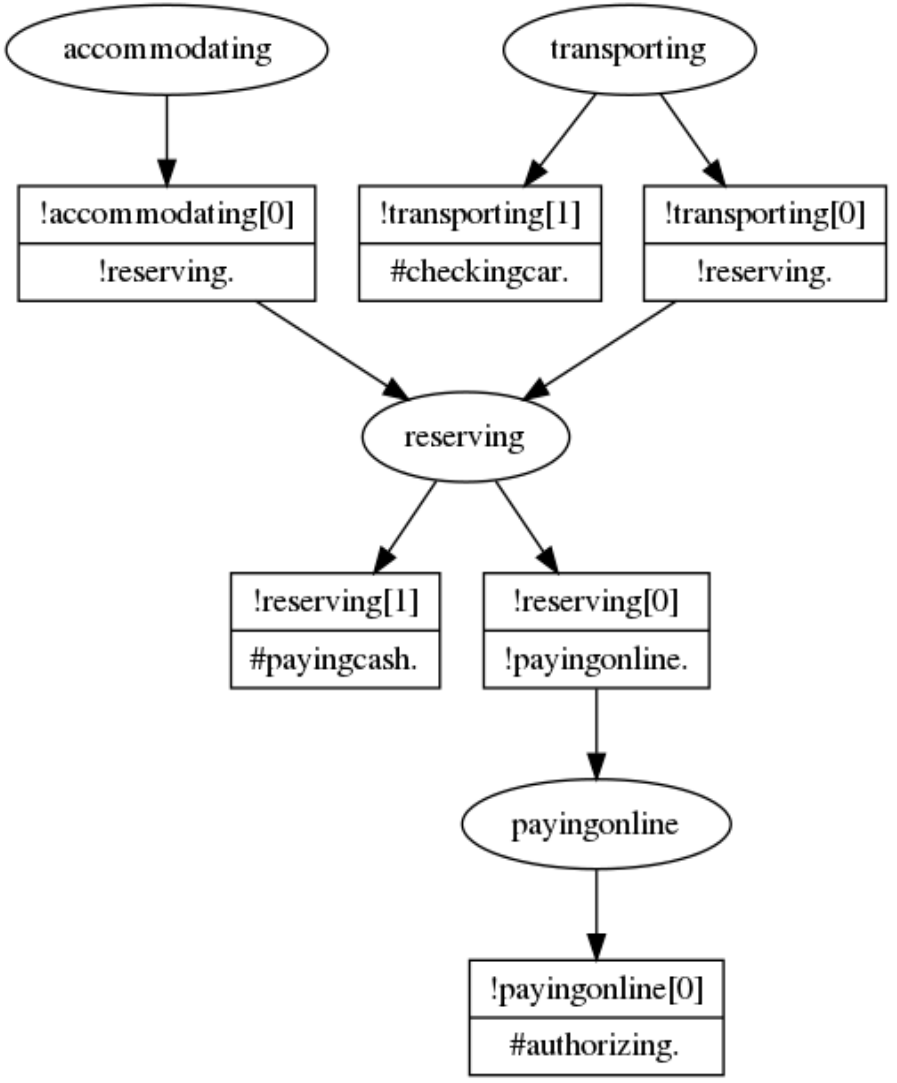
  e: invocation event, goal

  c: context condition

  p: plan body, sequence of steps

```
+!transporting : money =>
   !reserving.
+!transporting : car =>
   #checking_car.
```

```
+!transporting =>
   !reserving.
+!transporting =>
   #checking_car.
+!accommodating =>
   !reserving.
+!reserving =>
   !paying_online.
+!reserving =>
   #paying_cash.
+!paying_online =>
   #authorizing.
```



Applicability vs Preferability

- **Applicability**: *Which plan is applicable?*
- **Preferability**: *Which plan is preferred?*

- Current BDI frameworks
  - only express applicability
  - preference is implicit and static
  - through sequential ordering
  - resides in the designer's mind

```
+!transporting : money =>
   !reserving.
+!transporting : car =>
   #checking_car.
```

Outline

- Motivation
- BDI agents’ procedural knowledge
- Explicit preferences language: CP-Nets
- Reactivity as plan selection without reflection

CP-nets

- Compact representation language [Boutilier2004]:
  - conditional preferences
  - based on "all else being equal" (ceteris paribus) assumption

“I prefer to fly rather than to drive” + all else being equal

“if I’m flying I prefer night time, but if I’m driving I prefer day time” + all else being equal

conditioning preferences

CP-nets specification

```
not !transporting > !transporting : true.
not !reserving > !reserving : !transporting .
!reserving > not !reserving : not !transporting .
!accommodating > not !accommodating : !reserving.
not !accommodating > !accommodating : not !reserving.
!paying_online > not !paying_online : !transporting.
not !paying_online > !paying_online : not !transporting.
```

(potentially) strict ordering between outcomes

!travelling, !reserving,
not !accommodating, !paying online

>

!travelling, !reserving,
not !accommodating, not !paying online

Outline

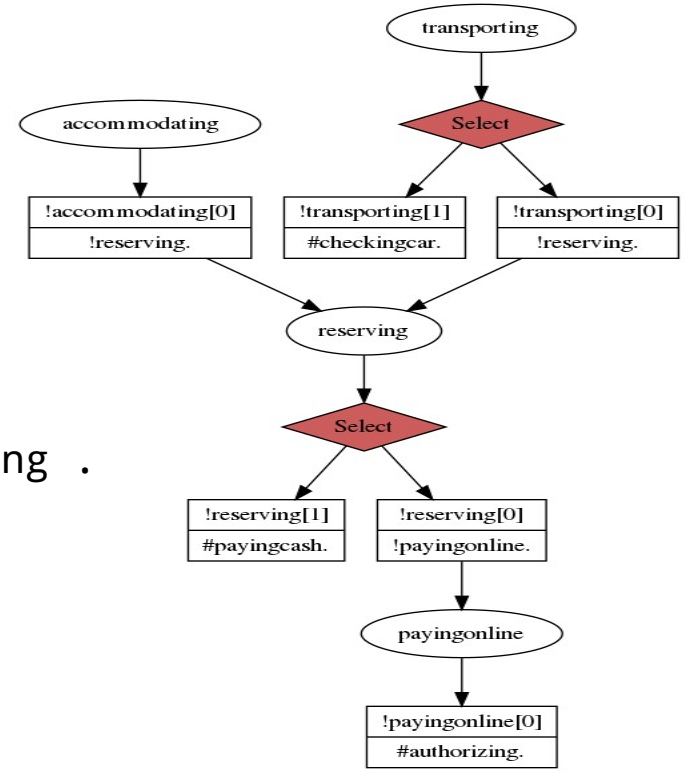
- Motivation
- BDI agents' procedural knowledge
- Explicit preferences language: CP-Nets
- Reactivity as plan selection without reflection

Reflection vs Reaction

- *What do we mean by reflection and reaction?*
- Reflection as ***problem-solving***
  - In cognitive terms: Looking inside one's own mind (**introspection**) and adapt own's behaviour to achieve the best possible results amongst the available choices (**adaptation**)
  - In computational terms: *introspection* of the program at ***run-time*** and ***online*** *behavioural/structural adaptation* of the program to obtain certain results

Reflection: preferences as a *rationale*

```
not !transporting > !transporting : true.
not !reserving > !reserving : !transporting .
!reserving > not !reserving : not !transporting .
!accommodating > not !accommodating : !reserving.
not !accommodating > !accommodating : not !reserving.
!paying_online > not !paying_online : !transporting .
not !paying_online > !paying_online : not !transporting .
```



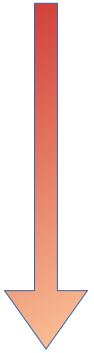
At run-time

stop and reflect by means of preferences at every <Select>

What is the best solution amongst the available ones?

Reactive choices for plan selection

- Selection paths are decided ***off-line***
  - requires known preferability between choices
  - preferabililty of choices should be computed based on explicit preferences

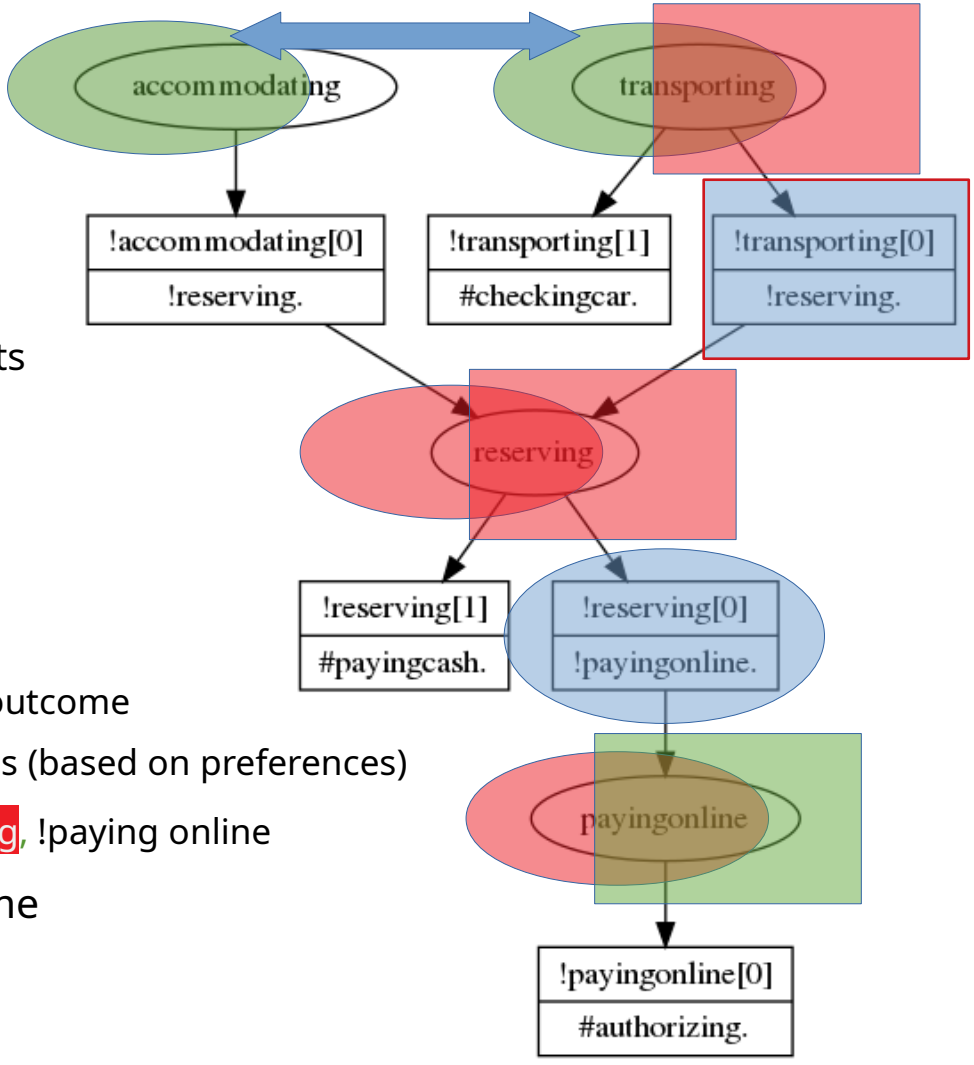


```
+!transporting : money =>
   !reserving.
+!transporting : car =>
   #checking_car.
```

Ordering algorithm

- For each plan g_i
  - Find all **certain** and **possible** past and future contexts
- Possible past contexts become conditions $C(g_i)$
- For each condition, $c \in C(g_i)$
  - Find the best possible outcome $o(c, g_i)$
    - Add adoption all goals that are members of c
    - Add adoption of certain past and future goals to the outcome
    - Add most optimistic state of the uncertain future goals (based on preferences)

  e.g. !reserving[0]: !travelling, !reserving, not !accommodating, !paying online
- Sort plans from best to worst based on their outcome



Sample results

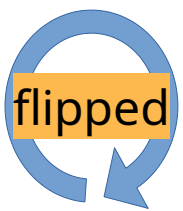
Non-Prioritized procedural knowledge

```
+!transporting => !reserving.
+!transporting => #checking_car.
+!reserving => !paying_online.
+!reserving => #paying_cash.
```

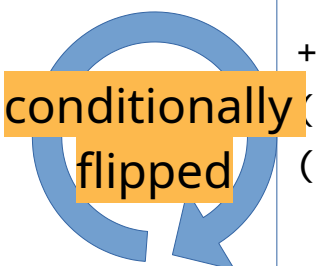
CP-net preferences

```
not !transporting > !transporting : true.
not !reserving > !reserving : !transporting.
...
```

Prioritized procedural knowledge
(~ AgentSpeak/LightJason)




```
+!transporting <=
    #checkingcar.
+!transporting <=
    !reserving.
```




```
+!reserving :
!trans. & not !accommodating
    <= !paying_online.
    <= #paying_cash.

+!reserving :
(not !trans. & !accommodating) |
(not !trans. & not !accommodating)
    <= #paying_cash.
    <= !paying_online.
```

Future directions

- Going beyond ***procedural***, ***propositional*** preferences
  - declarative (achievement) and maintenance goals
  - *first order logic* (FOL) specifications
- Going beyond ***plan selection***
  - Triggering event selection
  - Intention selection

Thank you!

Questions?