

User Understanding of Cognitive Processes in Simulation: A Tool For Exploring and Modifying

David Scerri, Sarah Hickmott, Lin Padgham

RMIT University, Melbourne, Australia

Outline

- 1 Motivation
- 2 Belief Desire Intention (BDI) Agents
- 3 Tool for Dynamic Exploration
- 4 Refining the Model
- 5 Discussion and Conclusion

Outline

- 1 Motivation
- 2 Belief Desire Intention (BDI) Agents
- 3 Tool for Dynamic Exploration
- 4 Refining the Model
- 5 Discussion and Conclusion

Representation of Agent Decisions and Behaviours

- Simple reactive rules work well for things like flocking, traffic movement, or crowd behaviour.
- But social science simulations often need more complex human behaviours.
- e.g. modelling behaviours and decisions in bushfire evacuation, or in a flood situation, will involve more complex decisions.
- Can be coded in java/python/etc. or represented as a FSM - but this is not easy for non-programmers to understand.
- We want something more intuitive for non-programmers.

Modelling of Human Behaviour

- Humans are **reactive** - but **not entirely**.
- They typically have **goals and plans** that extend over a **period of time**.
- They make and **adjust decisions** based on the **unfolding situation**.
- They **know what** they have been doing **and why** - this is part of what they do next.
- The **BDI agent paradigm** captures these aspects well and has been used extensively for developing intelligent agent systems.

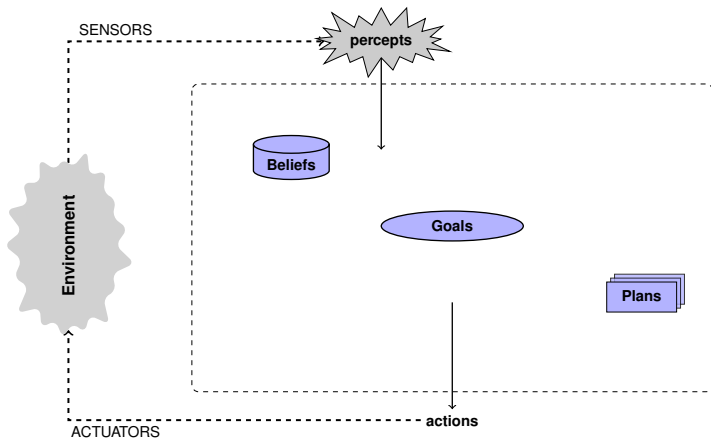
Outline

- 1 Motivation
- 2 Belief Desire Intention (BDI) Agents**
- 3 Tool for Dynamic Exploration
- 4 Refining the Model
- 5 Discussion and Conclusion

BDI Agent Oriented Programming

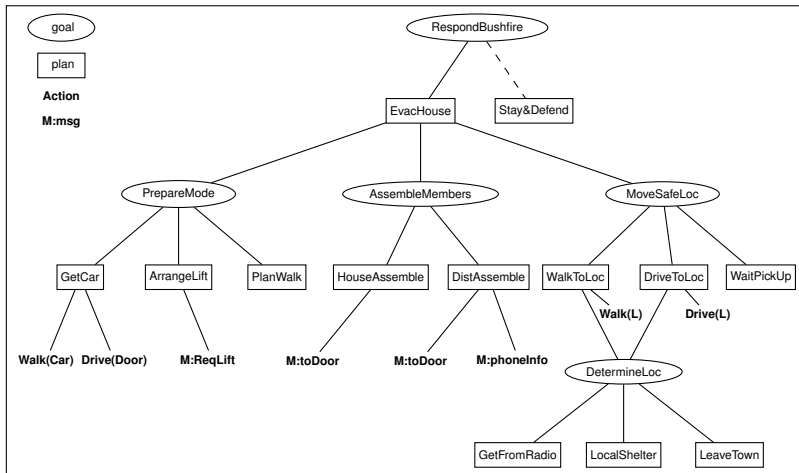
- BDI Agent-Oriented Programming provides abstraction at the level of **mental attitudes** to explain the operation of a system. **Beliefs, Desires, Intentions.**
- The **modularity of plans** makes it easy to develop complexity **incrementally.**
- The **goal oriented** approach makes it suitable for use in **dynamic environments.**
- Many **efficient and powerful** development environments available. **JACK, Jadex, Jason, PRS, 2APL, ...**
- BDI agent programs are **fast to develop.** A 2006 study showed:
 - Gain compared to Java programming **500%.**

Belief-Desire-Intention (BDI) Agent Architecture

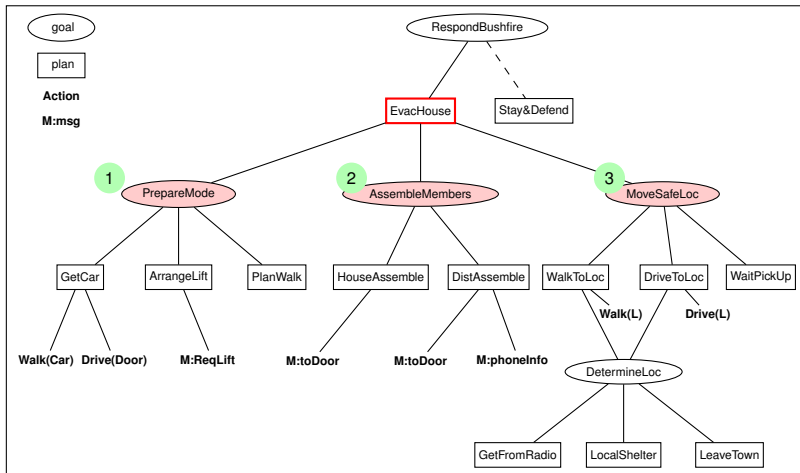


Percepts in, actions out. Internally, beliefs, goals and plans.

Example Plan Structure

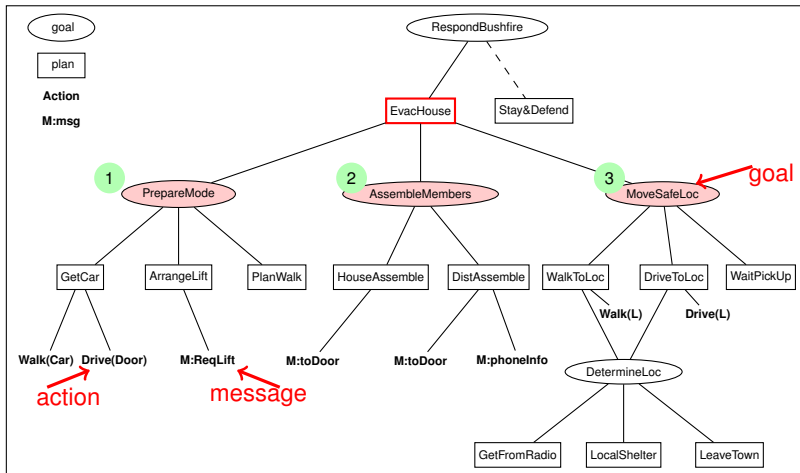


Example Plan Structure



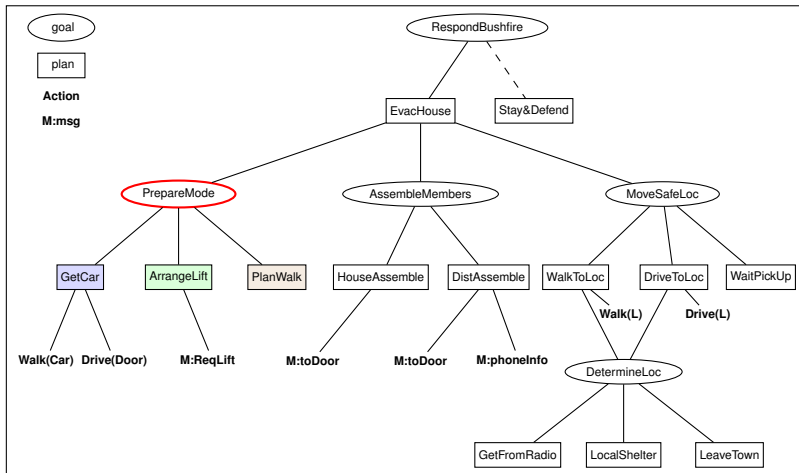
A plan is a **sequence of steps**

Example Plan Structure



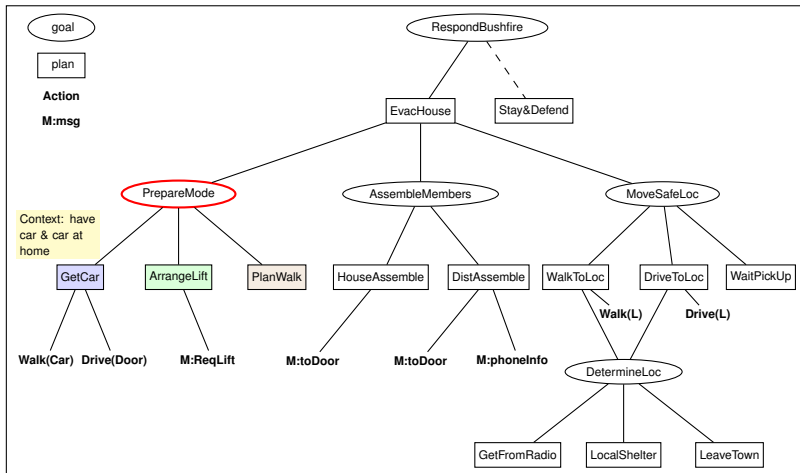
A step can be a **goal**, an **action**, a **message** to another agent, or some **computation**.

Example Plan Structure



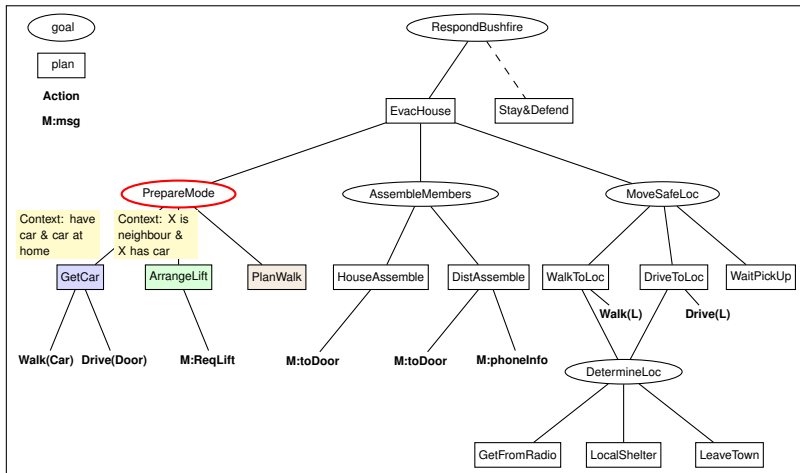
A goal may have **different plans**, for achieving it in **different situations**.

Example Plan Structure



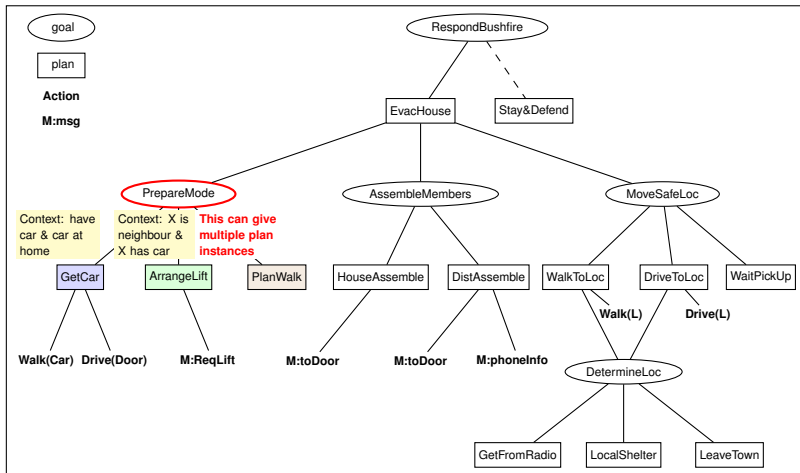
A goal may have **different plans**, for achieving it in **different situations**.

Example Plan Structure



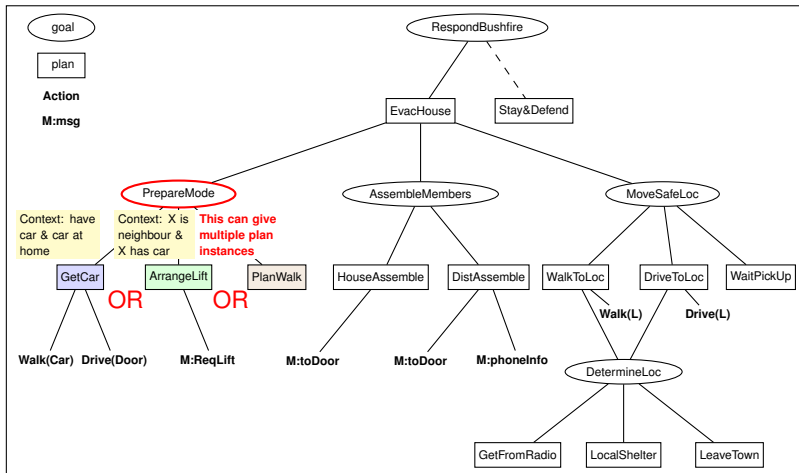
A goal may have **different plans**, for achieving it in **different situations**.

Example Plan Structure



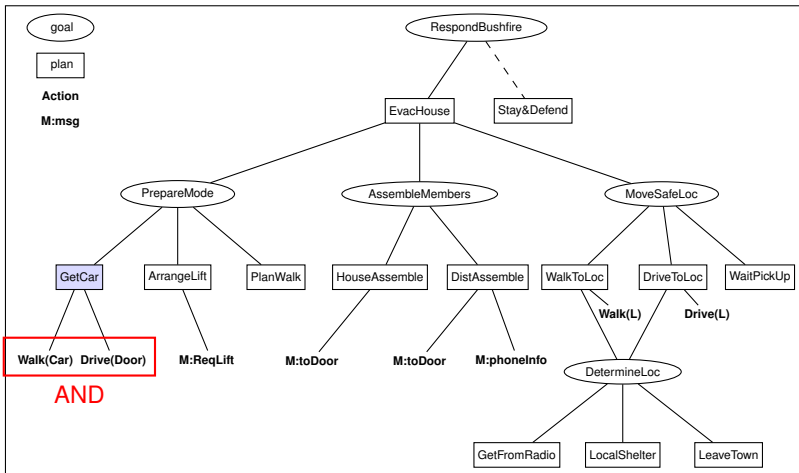
A goal may have **different plans**, for achieving it in **different situations**.

Example Plan Structure



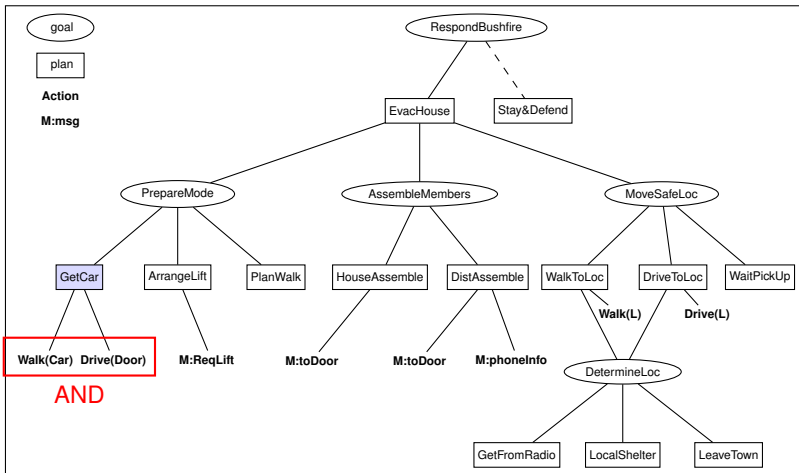
For a goal to succeed **one of the plans** must succeed. If one fails **try another**.

Example Plan Structure



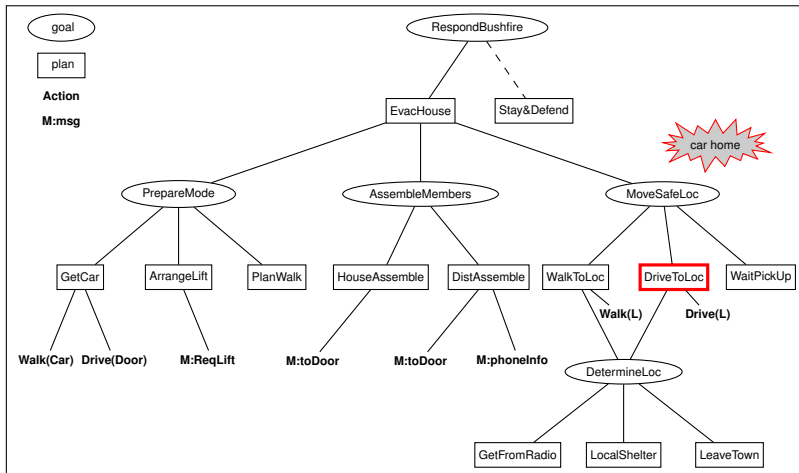
For a plan to succeed, **all** steps must succeed.

Example Plan Structure



If things fail, **recovery** happens as locally as possible

Example Plan Structure



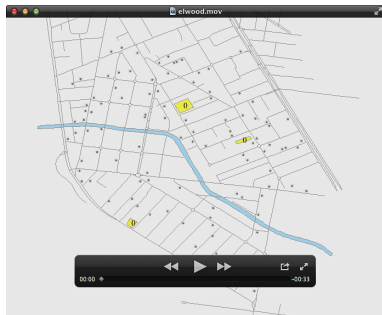
Plan selection **responsive** to changing environment.

Outline

- 1 Motivation
- 2 Belief Desire Intention (BDI) Agents
- 3 Tool for Dynamic Exploration**
- 4 Refining the Model
- 5 Discussion and Conclusion

Need to Understand at Runtime

- For **community use**: individuals want to identify with an agent, understand and maybe control it.
- For **participatory modelling**: experts need to understand what is happening, verify or change the model.
- **General debugging.**



Tool Overview

Three main ways the tool can be used:

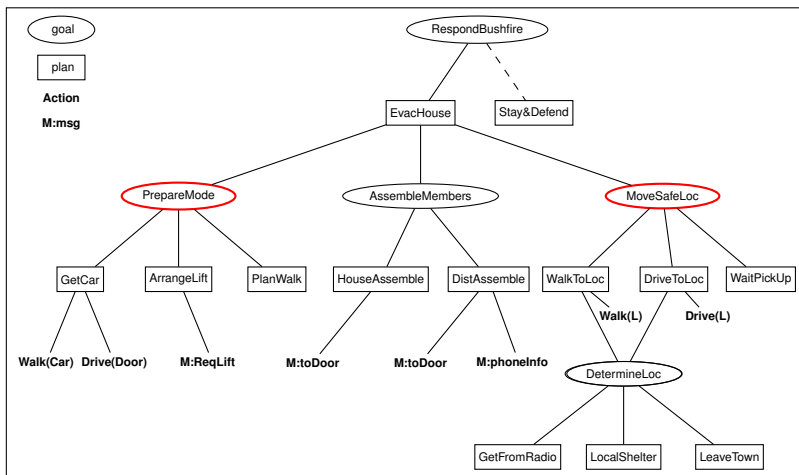
- To stop at a specific point, and explore what the agent has decided and why.
- To control the choices an agent makes at particular points, in order to observe the effect of those choices.
- To modify and refine the model - useful for participatory modelling.

Specify Stop Points for an Agent

Select **which agent** to track

Specify Stop Points for an Agent

Select **which agent** to track



Select **goals/decision points** to investigate.

Tool Information at Stop Point

The screenshot displays the 'Plan Selection' window of the BDI Interface Tool. It is divided into several sections:

- Selection Options:** Includes radio buttons for 'System Selection' (selected) and 'User Selection', and a 'Continue' button.
- Goal:** Shows the goal 'PrepareMode' with the description 'Arrange a mode of transport for reaching the evacuation area'.
- Hierarchy:** A tree diagram showing 'PrepareMode' at the top, branching into 'GetCar', 'ArrangeLift', and 'PlanWal'. 'GetCar' further branches into 'lk(Car, Drive(Door))' and 'M:F'. A red circle highlights the 'PrepareMode' node and its immediate children.
- Plans:**
 - Applicable Plans:** Lists '- ArrangeLift (3 instances) - 2' (highlighted) with sub-entries '(John, ABC123)', '(Sue, PQR 567)', and '(Steve, WXY 437)', and '+ PlanWalk (1 instance) - 3'.
 - Non-applicable Plans:** Lists 'GetCar - 1'.
 - Context clauses:** Lists 'HasCar(\$N, \$Car)', 'AtHome(\$Car)', and 'Neighbour(\$N)'.
- Details:**
 - Plan Description:** 'Prepare by organising to travel in a neighbour's car.' and 'Steps: Call neighbour, Request Lift, if Agreed, Arrange meeting point, if Not Agreed, fail'.
 - Parameter Bindings:** '\$N = John', '\$Car = ABC123', and 'Neighbour(\$N) & HasCar(\$N, \$Car)'.
 - Comments:** An empty text area.

The Goal-Plan hierarchy

Tool Information at Stop Point

The screenshot displays the 'Plan Selection' tool interface. The 'Goal' section, which is circled in red, contains the text: **PrepareMode**
Arrange a mode of transport for reaching the evacuation area.

The 'Selection Options' section includes 'System Selection' (selected), 'User Selection', and a 'Continue' button.

The 'Hierarchy' section shows a tree structure with 'PrepareMode' at the top, branching into 'GetCar', 'ArrangeLift', and 'PlanWal'. 'GetCar' further branches into 'lk(Car)Drive(Door)' and 'M:F'. A small diagram is visible at the bottom right of the hierarchy.

The 'Plans' section lists 'Applicable Plans':
- ArrangeLift (3 instances) - 2 (John, ABC123) (Sue, PQR 567) (Steve, WXY 437)
+ PlanWalk (1 instance) - 3

The 'Non-applicable Plans' section lists 'GetCar - 1'.

The 'Context clauses' section lists: HasCar(\$N, \$Car), AtHome(\$Car), and Neighbour(\$N).

The 'Details' section includes:
Plan Description: Prepare by organising to travel in a neighbour's car.
Steps: Call neighbour, Request Lift, if Agreed, Arrange meeting point, if Not Agreed, fail.
Parameter Bindings: \$N = John, \$Car = ABC123.
Neighbour(\$N) & HasCar(\$N, \$Car)

The 'Comments' section is empty.

The Goal **description**

Tool Information at Stop Point

The screenshot displays the 'Plan Selection' window of the BDI Interface Tool. The window is divided into several panels:

- Selection Options:** Includes radio buttons for 'System Selection' (selected) and 'User Selection', and a 'Continue' button.
- Goal:** Shows the goal 'PrepareMode' with the description 'Arrange a mode of transport for reaching the evacuation area'.
- Hierarchy:** A tree diagram showing 'PrepareMode' as the root, branching into 'GetCar', 'ArrangeLift', and 'PlanWalk'. 'GetCar' further branches into 'lk(Car)Drive(Door)' and 'M:F'. 'ArrangeLift' branches into a detailed sub-hierarchy of actions.
- Plans:** A list of applicable plans:
 - ArrangeLift (3 instances) - 2
 - (John, ABC123)
 - (Sue, PQR 567)
 - (Steve, WXY 437)
 - + PlanWalk (1 instance) - 3
- Non-applicable Plans:** Lists 'GetCar - 1'.
- Context clauses:** Lists 'HasCar(\$N, \$Car)', 'AtHome(\$Car)', and 'Neighbour(\$N)'.
- Details:**
 - Plan Description:** 'Prepare by organising to travel in a neighbour's car.'
 - Steps: Call neighbour, Request Lift, if Agreed, Arrange meeting point, if Not Agreed, fail.
 - Parameter Bindings:** '\$N = John', '\$Car = ABC123'.
 - Neighbour(\$N) & HasCar(\$N, \$Car)** (highlighted in green).
- Comments:** An empty text area.

The Applicable Plans

Tool Information at Stop Point

The screenshot displays the 'Plan Selection' window of the BDI Interface Tool. It is divided into several sections:

- Selection Options:** Includes radio buttons for 'System Selection' (selected) and 'User Selection', and a 'Continue' button.
- Goal:** Displays 'PrepareMode' with the description: 'Arrange a mode of transport for reaching the evacuation area'.
- Hierarchy:** A tree diagram showing 'PrepareMode' at the top, branching into 'GetCar', 'ArrangeLift', and 'PlanWal'. 'GetCar' further branches into 'lk(Car)Drive(Door)' and 'M:F'. A small thumbnail image is visible at the bottom right of the hierarchy.
- Plans:**
 - Applicable Plans:** Lists '- ArrangeLift (3 instances) - 2' (with sub-items '(John, ABC123)', '(Sue, PQR 567)', '(Steve, WXY 437)') and '+ PlanWalk (1 instance) - 3'. The 'ArrangeLift' plan is highlighted in blue.
 - Non-applicable Plans:** Lists 'GetCar - 1'.
 - Context clauses:** Lists 'HasCar(\$N, \$Car)', 'AtHome(\$Car)', and 'Neighbour(\$N)'.
- Details:**
 - Plan Description:** 'Prepare by organising to travel in a neighbour's car.' followed by steps: 'Call neighbour', 'Request Lift', 'if Agreed', 'Arrange meeting point', 'if Not Agreed', 'fail'.
 - Parameter Bindings:** Shows '\$N = John' and '\$Car = ABC123'. The binding 'Neighbour(\$N) & HasCar(\$N, \$Car)' is highlighted with a red circle.
 - Comments:** An empty text area.

The **Parameter Bindings** for the selected plan

Tool Information at Stop Point

The screenshot displays the 'Plan Selection' tool interface. It is divided into several panels:

- Selection Options:** Includes radio buttons for 'System Selection' (selected) and 'User Selection', and a 'Continue' button.
- Goal:** Displays the goal 'PrepareMode' with the description 'Arrange a mode of transport for reaching the evacuation area'.
- Hierarchy:** A tree diagram showing 'PrepareMode' at the top, branching into 'GetCar', 'ArrangeLift', and 'PlanWalk'. 'GetCar' further branches into 'lk(Car)Drive(Door)'. 'ArrangeLift' branches into 'M:F' and a detailed sub-hierarchy diagram.
- Plans:**
 - Applicable Plans:** Lists '- ArrangeLift (3 instances) - 2' (highlighted in blue) with parameters '(John, ABC123)', '(Sue, PQR 567)', and '(Steve, WXY 437)', and '+ PlanWalk (1 instance) - 3'.
 - Non-applicable Plans:** Lists 'GetCar - 1'.
- Context clauses:** Lists 'HasCar(\$N, \$Car)', 'AtHome(\$Car)', and 'Neighbour(\$N)'.
- Details:**
 - Plan Description:** Contains the text 'Prepare by organising to travel in a neighbour's car.' and a list of steps: 'Call neighbour', 'Request Lift', 'if Agreed', 'Arrange meeting point', 'if Not Agreed', and 'fail'. This section is circled in red.
 - Parameter Bindings:** Shows '\$N = John' and '\$Car = ABC123'.
 - Neighbour(\$N) & HasCar(\$N, \$Car):** A green text label.
- Comments:** An empty text area at the bottom.

The Plan Description

Exploring Non-Applicable Plans

The screenshot shows the 'Plan Selection' tool interface. It is divided into several sections:

- Selection Options:** Includes radio buttons for 'System Selection' (selected) and 'User Selection', and a 'Continue' button.
- Goal:** Displays the goal: 'Prepare a mode of transport for reaching the evacuation area' under the heading 'PrepareMode'.
- Hierarchy:** A tree diagram showing 'PrepareMode' at the top, branching into 'GetCar', 'ArrangeLift', and 'PlanWalk'. 'GetCar' further branches into 'lk(Car)Drive(Door)' and 'M:F'. A small thumbnail image is visible at the bottom right of the hierarchy.
- Plans:**
 - Applicable Plans:** Lists '- ArrangeLift (3 instances) - 2' (with sub-items for John, Sue, and Steve) and '+ PlanWalk (1 instance) - 3'.
 - Non-applicable Plans:** A list containing 'GetCar - 1', which is highlighted with a red circle.
 - Content clauses:** Lists 'HasCar(\$N, \$Car)', 'AtHome(\$Car)', and 'Neighbour(\$N)'.
 - Comments:** An empty text area.
- Details:**
 - Plan Description:** 'Prepare by organising to travel in own car.' and 'Steps: Walk to the parked car, Drive it to the front door of the house'.
 - Parameter Bindings:** Shows 'HasCar(\$N, \$Car) & AtHome(\$Car)'.

Can also **select** and get information about non-applicable plans

Exploring Non-Applicable Plans

The screenshot shows the Plan Selection tool interface with the following components:

- Selection Options:** System Selection (selected), User Selection, and a Continue button.
- Goal:** PrepareMode: Arrange a mode of transport for reaching the evacuation area.
- Hierarchy:** A tree diagram with 'PrepareMode' at the root, branching into 'GetCar', 'ArrangeLift', and 'PlanWalk'. 'GetCar' further branches into 'lk(Car)Drive(Door)' and 'M:F'. A small thumbnail diagram is visible at the bottom right of the hierarchy.
- Plan Selection:**
 - Plans:**
 - Applicable Plans:
 - ArrangeLift (3 instances) - 2 (John, ABC123) (Sue, PQR 567) (Steve, WXY 437) (Failed)
 - + PlanWalk (1 instance) - 3
 - Non-applicable Plans: GetCar - 1 (highlighted in blue)
 - Context clauses:** HasCar(\$N, \$Car), AtHome(\$Car), Neighbour(\$N)
 - Details:**
 - Plan Description:** Prepare by organising to travel in own car. Steps: Walk to the parked car, Drive it to the front door of the house.
 - Parameter Bindings:** HasCar(\$N, \$Car) & AtHome(\$Car) (circled in red)
 - Comments:** (Empty text area)

No parameter bindings. Also context failed (red).

Exploring Non-Applicable Plans

The screenshot shows the Plan Selection tool interface with the following components:

- Selection Options:** System Selection (selected), User Selection, and a Continue button.
- Goal:** PrepareMode: Arrange a mode of transport for reaching the evacuation area.
- Hierarchy:** A tree diagram with PrepareMode at the root, branching into GetCar, ArrangeLift, and PlanWal. GetCar further branches into lk(Car)Drive(Door) and M:F.
- Plan Selection:**
 - Applicable Plans:**
 - ArrangeLift (3 instances) - 2 (John, ABC123) (Sue, PQR 567) (Steve, WXY 437) (Failed)
 - + PlanWalk (1 instance) - 3
 - Non-applicable Plans:** GetCar - 1 (highlighted in blue)
 - Context clauses:**
 - HasCar(\$N, \$Car)
 - AtHome(\$Car)
 - Neighbour(\$N)
 - Details:**
 - Plan Description:** Prepare by organising to travel in own car. Steps: Walk to the parked car, Drive it to the front door of the house.
 - Parameter Bindings:** (Empty)
 - Context clauses:** HasCar(\$N, \$Car) & AtHome(\$Car) (circled in red)
 - Comments:** (Empty)

In this example the first clause could be satisfied (green), but not the second (red)

Further Information

The screenshot displays the 'Plan Selection' tool interface. It is divided into several sections:

- Selection Options:** Includes radio buttons for 'System Selection' (selected) and 'User Selection', and a 'Continue' button.
- Goal:** Displays the goal 'PrepareMode' with the description 'Arrange a mode of transport for reaching the evacuation area'.
- Hierarchy:** A tree diagram showing 'PrepareMode' at the top, branching into 'GetCar', 'ArrangeLift', and 'PlanWal'. 'GetCar' further branches into 'lk(Car)Drive(Door)' and 'M:F'. A small diagram is visible at the bottom right of the hierarchy.
- Plans:**
 - Applicable Plans:** Lists '- ArrangeLift (3 instances) - 2' (with sub-items '(John, ABC123)' and '(Sue, PQR 567) (Steve, WXY 437) (Failed)') and '+ PlanWalk (1 instance) - 3'.
 - Non-applicable Plans:** Lists 'GetCar - 1'.
 - Context clauses:** Lists 'HasCar(\$N, \$Car)', 'AtHome(\$Car)', and 'Neighbour(\$N)'. This section is circled in red.
- Details:**
 - Plan Description:** 'Prepare by organising to travel in own car.' and 'Steps: Walk to the parked car, Drive it to the front door of the house'.
 - Parameter Bindings:** Shows 'HasCar(\$N, \$Car) & AtHome(\$Car)'.
 - Comments:** An empty text area.

The **Context clauses** used for any of the plans

Further Information

The screenshot displays the 'Plan Selection' tool interface, which is organized into several panels:

- Selection Options:** Contains radio buttons for 'System Selection' (selected) and 'User Selection', along with a 'Continue' button.
- Goal:** Displays the goal 'PrepareMode' with the description 'Arrange a mode of transport for reaching the evacuation area'.
- Hierarchy:** A tree diagram showing 'PrepareMode' at the top, branching into 'GetCar', 'ArrangeLift', and 'PlanWalk'. 'GetCar' further branches into 'lk(Car)Drive(Door)' and 'M:F'. A small thumbnail diagram is visible at the bottom right of this panel.
- Plans:**
 - Applicable Plans:** Lists three plans with their priorities: '- ArrangeLift (3 instances) - 2' (with a red circle around the priority 2), '(John, ABC123) (Sue, PQR 567) (Steve, WXY 4567) (Failed)', and '+ PlanWalk (1 instance) - 3' (with a red circle around the priority 3).
 - Non-applicable Plans:** Lists 'GetCar - 1' (with a red circle around the priority 1), which is highlighted in blue.
- Context clauses:** Lists 'HasCar(\$N, \$Car)', 'AtHome(\$Car)', and 'Neighbour(\$N)'.
- Details:**
 - Plan Description:** 'Prepare by organising to travel in own car.' and 'Steps: Walk to the parked car, Drive it to the front door of the house'.
 - Parameter Bindings:** Shows 'HasCar(\$N, \$Car) & AtHome(\$Car)'.
 - Comments:** An empty text area.

The numbers indicate **plan priority**

Further Information

The screenshot shows the Plan Selection tool interface. The main window is titled "Plan Selection" and contains several panels:

- Selection Options:** Includes radio buttons for "System Selection" (selected) and "User Selection", and a "Continue" button.
- Goal:** Displays the goal "PrepareMode" with the description "Arrange a mode of transport for reaching the evacuation area".
- Hierarchy:** A tree diagram showing "PrepareMode" as the root, branching into "GetCar", "ArrangeLift", and "PlanWalk". "GetCar" further branches into "lk(Car)Drive(Door)" and "M:F".
- Plans:**
 - Applicable Plans:**
 - ArrangeLift (3 instances) - 2
 - (John, ABC123)
 - (Sue, PQR 567)
 - (Steve, WXY 437) (Failed)
 - + PlanWalk (1 instance) - 3
 - Non-applicable Plans:** GetCar - 1
 - Context clauses:** HasCar(\$N, \$Car), AtHome(\$Car), Neighbour(\$N)
 - Comments:** A text area for user comments.
- Details:**
 - Plan Description:** Prepare by organising to travel in a neighbour's car. Steps: Call neighbour, Request Lift, if Agreed: Arrange meeting point, if Not Agreed: fail.
 - Parameter Bindings:** \$N = John, \$Car = ABC123. A note "Neighbour(\$N) & HasCar(\$N, \$Car)" is shown in green.

Any instances which have been tried and **failed** for this goal instance are annotated

Interactive Plan Selection

The screenshot displays the 'Plan Selection' tool interface. It is divided into several sections:

- Selection Options:** Includes radio buttons for 'System Selection' and 'User Selection' (which is selected), and a 'Continue' button.
- Goal:** Labeled 'PrepareMode', with the description 'Arrange a mode of transport for reaching the evacuation area'.
- Hierarchy:** A tree diagram showing 'PrepareMode' at the top, branching into 'GetCar', 'ArrangeLift', and 'PlanWalk'. 'GetCar' further branches into 'lk(Car)Drive(Door)' and 'M:F'. A small thumbnail diagram is visible at the bottom right of this section.
- Plans:** A list of 'Applicable Plans' where '+ PlanWalk (1 instance) - 3' is highlighted with a blue background and a red circle. Above it are three instances of 'ArrangeLift' (2 successful, 1 failed). Below are 'Non-applicable Plans' (e.g., 'GetCar - 1').
- Context clauses:** A list of conditions: 'HasCar(\$N, \$Car)', 'AtHome(\$Car)', and 'Neighbour(\$N)'.
- Details:** Includes a 'Plan Description' (text area), 'Parameter Bindings' (text area), and 'Comments' (text area). The word 'True' is displayed in green below the parameter bindings area.

Selected **PlanWalk**

Interactive Plan Selection

The screenshot displays the 'Plan Selection' tool interface. On the left, the 'Selection Options' panel has 'User Selection' selected with a red circle around it, and a 'Continue' button below it. The 'Goal' panel shows the goal: 'PrepareMode: Arrange a mode of transport for reaching the evacuation area'. The 'Hierarchy' panel shows a tree structure with 'PrepareMode' at the root, branching into 'GetCar', 'ArrangeLift', and 'PlanWalk'. 'GetCar' has sub-nodes 'lk(Car)Drive(Door)' and 'M:F'. 'ArrangeLift' has a sub-node 'M:F'. 'PlanWalk' has a sub-node 'M:F'. The 'Plans' panel lists 'Applicable Plans' with three entries: 'ArrangeLift (3 instances) - 2' (with sub-entries for John, Sue, and Steve), 'PlanWalk (1 instance) - 3' (highlighted in blue), and 'GetCar - 1'. The 'Context clauses' panel lists 'HasCar(\$N, \$Car)', 'AtHome(\$Car)', and 'Neighbour(\$N)'. The 'Details' panel shows 'Plan Description' and 'Parameter Bindings'. The 'Comments' panel is empty.

User Selection

Select Now or Always

When the user makes a different selection than the system would have made, this can be for two different reasons:

- To explore what would happen with a different selection. This does not require a permanent change to the program.
- To refine the program to make this choice in future. Requires more information and change to the program.

E.g. Select “PlanWalk”; need to know whether this choice should be incorporated into the program, and if so under what conditions.

Outline

- 1 Motivation
- 2 Belief Desire Intention (BDI) Agents
- 3 Tool for Dynamic Exploration
- 4 Refining the Model**
- 5 Discussion and Conclusion

Modifying Selection for Future

Possibilities are:

- The selected plan should **always** be chosen if applicable. Modify priorities to reflect this.
- In **some situations** (including current one), this plan should be preferred. Describe in comments pane the situational factors that indicate that this should be the preferred plan.
- A **specific instance of a plan type** should be chosen. Priority is on the type, so same for all instances of a type. Describe in comments pane what factors about an instance are used to make this choice.

The screenshot displays the 'Plan Selection' window, which is divided into several sections:

- PrepareMode:** A text area on the left with the instruction: 'Change a mode of transport for reaching the evacuation area'. A yellow callout bubble points to a 'PlanWal' button below it.
- Plans:** A central list of plans. Under 'Applicable Plans', there are three entries:
 - ArrangeLift (3 instances) - 2 (John, ABC123) (Sue, PQR 567) (Steve, WXY 437) (Failed)
 - + PlanWalk (1 instance) - 3 (highlighted in blue)
- Non-applicable Plans:** A list containing 'GetCar - 1'.
- Context clauses:** A list containing 'HasCar(\$N, \$Car)', 'AtHome(\$Car)', and 'Neighbour(\$N)'. A green 'True' indicator is visible to the right.
- Comments:** A large empty text area at the bottom for user input.
- Detail:** A vertical pane on the right side, partially visible, showing 'Plan I' and 'Prep foot.'.
- Paran:** A vertical pane on the right side, partially visible.

At the bottom left, a small diagram shows a network of nodes and connections, with a yellow callout bubble pointing to a specific node.

Modifying Selection for Future

Possibilities are:

- The selected plan should **always** be chosen if applicable. Modify priorities to reflect this.
- In **some situations** (including current one), this plan should be preferred. Describe in comments pane the situational factors that indicate that this should be the preferred plan.
- A specific instance of a plan type should be chosen. Priority is on the type, so same for all instances of a type. Describe in comments pane what factors about an instance are used to make this choice.

The screenshot displays the 'Plan Selection' window. On the left, a 'PrepareMode' pane contains text about changing a mode of transport. Below it, a 'PlanWal' pane shows a tree diagram with a yellow highlight on a node. The main area is divided into three sections: 'Applicable Plans' (listing 'ArrangeLift' and 'PlanWalk', with 'PlanWalk' selected), 'Non-applicable Plans' (listing 'GetCar'), and 'Context clauses' (listing 'HasCar', 'AtHome', and 'Neighbour'). A 'Comments' pane at the bottom is circled in red. On the right, a 'Detail' pane shows 'Plan I' and 'Prep foot.', and a 'Param' pane shows 'True'.

Modifying Selection for Future

Possibilities are:

- The selected plan should **always** be chosen if applicable. Modify priorities to reflect this.
- In **some situations** (including current one), this plan should be preferred. Describe in comments pane the situational factors that indicate that this should be the preferred plan.
- A **specific instance of a plan type** should be chosen. Priority is on the type, so same for all instances of a type. Describe in comments pane what factors about an instance are used to make this choice.

The screenshot displays the 'Plan Selection' window. On the left, a 'PrepareMode' pane is partially visible. The main area is divided into several sections:

- Applicable Plans:** A list of plans with their instances and counts. The instance '(Sue, PQR 567)' is highlighted in blue.
- Non-applicable Plans:** A list of plans that are not currently applicable, such as 'GetCar - 1'.
- Context clauses:** A list of conditions like 'HasCar(\$N, \$Car)', 'AtHome(\$Car)', and 'Neighbour(\$N)'.
- Comments:** A text area at the bottom, which is circled in red in the image, intended for describing situational factors.

On the right side, a 'Detail' pane shows a 'Plan I' section with a 'Prej' field and a 'Detail' section with a 'Req' field. Below that, a 'Param' section shows '\$N = \$Car' and a 'Neig' field.

Modifying the Model

Ways to change the model:

- 1 change plan priorities (changing ranking)
- 2 change context condition of one or more plans (changing applicability)
 - may require new information from environment
- 3 provide extra selection information
- 4 provide additional plans

Modifying the Model

Ways to change the model:

- 1 change plan priorities (changing ranking)
- 2 change context condition of one or more plans (changing applicability)
 - may require new information from environment
- 3 provide extra selection information
- 4 provide additional plans

These **can be done directly** as long as no new information is required.

Modifying the Model

Ways to change the model:

- 1 change plan priorities (changing ranking)
- 2 change context condition of one or more plans (changing applicability)
- 3 provide extra selection information
- 4 provide additional plans

These **must be programmed**. Comments can be provided for developer.

Modifying Context: some issues

- Modifying context conditions must be done with care!
- If a clause that binds a variable is removed, this may cause a problem if variable is used in the plan.
 - E.g. Neighbour(\$N) in context;
Ring \$N in plan.
- Could reason about this and provide warnings or disallow. Not currently done.

Outline

- 1 Motivation
- 2 Belief Desire Intention (BDI) Agents
- 3 Tool for Dynamic Exploration
- 4 Refining the Model
- 5 Discussion and Conclusion**

Need to Consider All Plans for that Goal

- To understand a plan selection, need to understand what was not applicable.
- e.g. choose walk because nothing else available, not because of context for walk plan...
- Decisions leading to current point may also be relevant - this is evident in the hierarchy.

Need to Generalise Current State to Express Context

- Ideally allow user to indicate plan selection, determine (preference) context from current state.
- Not that simple.
 - some aspects are irrelevant;
 - some aspects must be generalised.
- Could potentially use learning
 - but simpler to just ask...

Conclusion

- A tool to understand the internals of a single agent.
- Useful for
 - community awareness tools: user identification,
 - debugging and development,
 - participatory modelling,
 - domain expert input.
- Initial development in JACK - now planning development in Jadex (open source).
- Many potential areas of development, but want to trial and evaluate first.

Questions

