TikZ AI Plans: Visualize Artificial Intelligence Plans with TikZ

In this tutorial, we will explore the process of using the TikZ package to generate "plans", which are the central concept within Automated Planning, a subdiscipline of Artificial Intelligence. More specifically, we introduce the TikZ library <aiplans>. It allows to specify a "domain model" for all actions (similar to the well-known planning domain definition language PDDL and its variant HDDL for hierarchical planning) which can then be used for the required diagrams. Diagrams depicting plans can be sequential (action sequences), but also allow them to be depicted in a partially ordered way, including causal links and ordering constraints as done in Partial Order Causal Link (POCL) plans. The library is intended for researchers and students, e.g., for presentations and scientific publications. An example is provided in Figure 1.

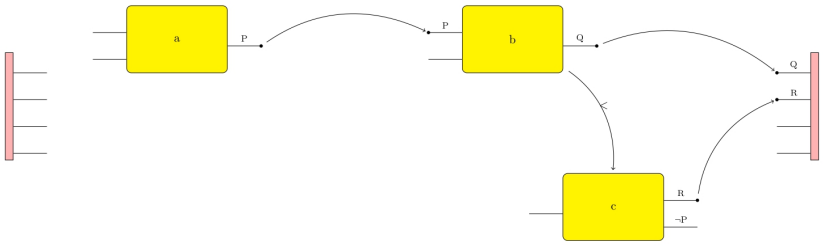

Figure 1: An example POCL plan involving both causal links (connecting action effects with another action's precondition) and an ordering constraint (between action b and c).

Copyright

Permission is granted to distribute and/or modify both the documentation and the code under the conditions of the LaTeX Project Public License, either version 1.3 of this license or (at your option) any later version. The latest version of this license is available at `https://ctan.org/license/lppl1.3`.

1 Set Up the LaTeX Project

1.1 Create a Folder Structure

Set up a directory with the following structure:

- `project_root/`
  - `domain.tex` (for defining TikZ styles)
  - `main.tex` (the main document that compiles everything)
  - `yourplan.tex`, such as `transportation.tex` (for example)

1.2 Main LaTeX Document (`main.tex`)

Create a `main.tex` file that includes the `domain.tex` and the specific plan file (e.g., `plan-transportation.tex`):

Main LaTeX Document (main.tex)

```
\documentclass[border=1in]{standalone}
\usepackage{../aiplan} % Include your custom TikZ style definitions
\begin{document}
\input{domain} % Load the TikZ style definitions
\input{yourplan} % Load the specific plan
\end{document}
```

Replace `yourplan` with the actual plan file name you are using.

2 Define Custom TikZ Styles (`domain.tex`)

In this step, you will create a `domain.tex` file where you define custom TikZ styles that represent different actions in your diagrams. These styles determine the appearance and behavior of the elements in your diagram, such as shapes, colors, and connections.

Before defining the styles, it is important to understand the structure of the style commands. Each TikZ style you define follows a specific pattern:

TikZ Style Definition Example

```
NameOfScheme /.style n args = {num of input parameters}
{
    action=
    % number of precs/effs
    {num of precondition lines (on the left side of action)}
    {num of effect lines (on the right side of action)}
    % prec/eff labels
    {sequence of precondition labels}
    {sequence of effect labels}
    % length of precs/effs
    {length of preconditions}
    {length of effects}
    % action name
```

```
        {action name}
        % prec/eff label position
        {precondition/effect label position: side or top}
    }
```

Example 1: Custom Styles for a Partial Order Causal Link (POCL) Plan (This code belongs to `domain.tex`)

In this example, the styles correspond to the actions like `a`, `b`, and `c`, with specific preconditions and effects.

1. Base Height Command

Base Height Command

```
\newcommand{\baseHeight}{0.8}
```

- This command defines a base height multiplier that helps adjust node sizes consistently throughout the diagram. Although it's not directly used in the following code, it is useful for maintaining uniformity when needed.

2. TikZ Style Definitions

TikZ Style Definitions

```
\tikzset{
```

- This line begins the `TikZ` set block where all the style definitions for nodes are declared.

a) Initial State (INIT) Node

INIT Node Style

```
INIT/.style={
    init={4}{ , , , }{1},
    fill = red
},
```

- `INIT`: This defines the initial state of the depicted plan. It represents the starting conditions before any actions have been performed.
- `Visual Style`: The node is filled with red, indicating that it is the starting state of the system.

b) Action 'a' Node

Action 'a' Node Style

```
a/.style n args ={1}{
    action={2}{1}{ , }{P}{1}{1}{a}{0},
    fill = yellow
},
```

- `Action 'a'`: This defines an action where certain preconditions need to be satisfied (P) before the action can be executed. The action has one effect, as represented by the P .
- `Visual Style`: The node for this action is filled with yellow to visually distinguish it from other types of nodes (e.g., the initial state).

c) Action 'b' Node

Action 'b' Node Style

```
b/.style n args ={1}{
    action={2}{1}{P,}{Q}{1}{1}{b}{0},
    fill = yellow
},
```

- `Action 'b'`: This defines an action with preconditions P that must be true before the action can execute, resulting in effect Q .
- `Visual Style`: Like a , this node is filled with yellow.

d) Action 'c' Node

Action 'c' Node Style

```
c/.style n args ={2}{
    action={1}{2}{ }{R,$\neg$P}{1}{1}{c}{0},
    fill = yellow
},
```

- `Action 'c'`: This defines an action with preconditions and two effects. One of the effects is R , and another is ¬P (the negation of P).
- `Visual Style`: Again, the action node is visually represented with yellow, consistent with the other action nodes.

e) Goal State (GOAL) Node

GOAL Node Style

```
GOAL/.style={
    goal={4}{Q,R, , }{1},
    fill = red
}
```

- `GOAL`: Defines the goal state node. It represents the final conditions to be achieved in the system. For example, in this case, the goals are conditions Q and R .
- `Visual Style`: The node is filled with red to highlight that it represents the goal state, making it stand out in the diagram.

```
}
```

- Ends the tikzset block, completing the definition of all the node styles.

Summary

- `INIT`: Represents the initial state.
- `GOAL`: Represents the goal state that the plan aims to achieve.
- `a`: Represents an action that occurs when certain preconditions are met and produces a specific effect.
- `b`: Represents another action with preconditions and effects.
- `c`: Represents an action with multiple effects, including negation

Example 2: Custom Styles for a Transportation Plan (This code belongs to `domain.tex`)

In this example, the styles correspond to actions like `LOAD`, `UNLOAD` and `MOVE` in a transportation scenario.

1. Base Height Command

Base Height Command

```
\newcommand{\baseHeight}{0.8}
```

- Defines a base height multiplier that can be used to adjust node sizes consistently throughout the diagram.

2. TikZ Style Definitions

TikZ Style Definitions

```
\tikzset{
```

- Begins the TikZ set block where all the style definitions for nodes are declared.

a) Initial State (INIT) Node

INIT Node Style

```
INIT/.style={
    init={5}{{at(R,L)},{at(R,B)},{at(A,L)},{at(B,L)},{fuel(R)}}{1.5},
    fill=red,
    minimum height=5cm,
    minimum width=1cm
},
```

- `INIT`: This defines the initial state of the system, representing the starting conditions before any actions have been performed.
- `Conditions`:

- `{5}` : This number indicates the total number of conditions that define the initial state of the system.
- `at(R,L)` : The robot (denoted by `R`) is at location `L`.
- `at(R,B)` : The robot `R` is also at location `B`.
- `at(A,L)` : Object `A` is at location `L`.
- `at(B,L)` : Object `B` is at location `L`.
- `fuel(R)` : The robot `R` has fuel.
- `1.5` : This parameter defines the vertical spacing between the conditions inside the node.

- `Visual Style`: The node is filled with red and given a minimum height of 5cm and a minimum width of 1cm to ensure it is clearly visible in the diagram.

b) Load Action Node

Load Action Node Style

```
Load/.style n args ={3}{
    action={2}{1}
    {at{(#3,#2)},at{(#1,#2)}}{in{(#1,#3)}}{1}{1}{Load(#1,#2,#3)}{0},
    fill=yellow,
    minimum height=2cm,
    minimum width=3cm
},
```

- `Load`: Defines an action with three arguments, where a robot (represented by `#3`) loads an object (represented by `#1`) at a certain location (represented by `#2`).
- `Preconditions`: Before the `Load` action can occur, both the robot and the object must be at the same location. This is indicated by the conditions `at(#3,#2)` and `at(#1,#2)`, meaning "the robot is at location `#2`" and "the object is at location `#2`".
- `Effects`: After the `Load` action, the object is now inside the robot. This is represented by `in(#1,#3)`, which means "the object is inside the robot".
- `Visual Style`: The node for this action is visually represented with a yellow fill color. The minimum height of the node is set to 2 cm, and the minimum width is set to 3 cm, ensuring it matches the size and style of other action nodes in the diagram.

c) Unload Action Node

Unload Action Node Style

```
Unload/.style n args ={3}{
    action={2}{1}
    {in{(#1,#3)},at{(#3,#2)}}{at{(#1,#2)}}{1}{1}{Unload(#1,#2,#3)}{0},
    fill=yellow,
    minimum height=2cm,
```

```
    minimum width=3cm
},
```

- `Unload`: Defines an action where a robot (represented by `#3`) unloads an object (represented by `#1`) at a certain location (represented by `#2`).
- `Preconditions`: Before the `Unload` action can occur, the object must be inside the robot. This is indicated by the condition `in(#1,#3)`, meaning "the object is inside the robot".
- `Effects`: After the `Unload` action, the object is now placed at the specified location. This is represented by `at(#1,#2)`, meaning "the object is now at location `#2`".
- `Visual Style`: The node for this action is also visually represented with a yellow fill color, maintaining consistency with the `Load` action node. The minimum height is set to 2 cm, and the minimum width is set to 3 cm.

d) Move Action Node

Move Action Node Style

```
Move/.style n args ={3}{
    action={2}{1}
    {at{(#3,#1)},fuel(#3)}{at{(#3,#2)}}{1}{1}{Move(#1,#2,#3)}{0},
    fill=yellow,
    minimum height=2cm,
    minimum width=3cm
},
```

- `Move`: Defines an action where a robot (represented by `#3`) moves from one location (represented by `1`) to another (represented by `#2`).
- `Preconditions`: The robot must be at the starting location with sufficient fuel to perform the move. This is indicated by the conditions `at(#3,#1)` (meaning "the robot is at the starting location") and `fuel(#3)` (meaning "the robot has enough fuel").
- `Effects`: After the `Move` action, the robot is now at the destination location. This is represented by `at(#3,#2)`, meaning "the robot is now at the destination".
- `Visual Style`: This node is visually represented with a yellow fill color, consistent with the other action nodes. The minimum height is set to 2 cm, and the minimum width is set to 3 cm.

e) Goal State (GOAL) Node

GOAL Node Style

```
GOAL/.style={
    goal={2}{{at(A,P)}, {at(B,P)}}{1.5},
    fill=red,
    minimum height=5cm,
    minimum width=1cm
}
```

- `GOAL`: Defines the goal state node, representing the final conditions to be achieved.
- `Conditions`: Objects A and B must both be at the final location P.
- `Visual Style`: The node is filled with red, similar to the `INIT` node, and sized to be prominently visible.

```
}
```

- Ends the block, completing the definition of all the node styles.

Summary

- `INIT`: Represents the initial state.
- `GOAL`: Represents the goal description the plan aims to achieve.
- `Load`: Represents the action of loading an object into the robot.
- `Unload`: Represents the action of unloading an object from the robot.
- `Move`: Represents the action of moving the robot from one location to another.

3 Create the Specific Plan File

In this step, you will create a plan file that generates the desired diagram. This file contains the actual TikZ commands needed to structure and visualize the POCL plan, demonstrating the sequence of actions and their dependencies from the initial state to the goal state.

Before we dive into the code, let's briefly discuss how the plan is drawn using TikZ:

- **Node Definition**: Nodes represent the different actions in the POCL plan. Each node is defined with specific attributes such as position, size, and color. These nodes are visually distinct to indicate different actions (e.g., `Load`, `Unload`, `Move`) and states (e.g., `INIT`, `GOAL`).
- **Causal Links**: After defining the nodes, the next step is to connect them with arrows, which are referred to as causal links in POCL diagrams. These arrows visually represent the dependencies between actions by connecting the preconditions of one action to the effects of another. This ensures that the sequence of actions follows logically from the initial state to the goal state.
- **Positioning and Layout**: The nodes and causal links are strategically placed to create a clear and understandable diagram. Coordinates are used to specify the position of each node, ensuring that the plan is organized and easy to follow.

Example 1 (`plan.tex` , This code belongs to `yourplan.tex`)

Introduction to Partial Order Causal Link (POCL) Plan

This example demonstrates the use of TikZ and custom styles to define a POCL (Partial Order Causal Link) plan diagram. POCL plans are used to represent a sequence of actions that are

partially ordered, with causal links illustrating the dependencies between actions. The goal is to show how the system transitions from the initial state to the goal state through a series of actions.

Code

a) Initial State Node Added

Initial State Node

```
\begin{tikzpicture}
    % Define the initial state node representing the starting conditions.
    % This node is placed at the coordinates (0, 0) in the diagram.
    \node[INIT, fill=red!30] (init) at (0,0){};
\end{tikzpicture}
```

- The initial state node (INIT) is defined and positioned at the origin (0,0) with a red fill. This represents the starting conditions of the scenario, as shown in Figure 2.

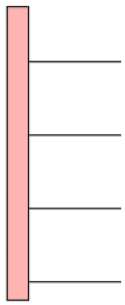

Figure 2: The initial state node in the plan, represented by INIT.

b) Action Nodes Added

Action Nodes

```
    % Define the action nodes for 'a', 'b', and 'c' actions.
    \node[a] (a) at (10,2) {};
    \node[b] (b) at (18,2) {};
    \node[c] (c) at (5,-3) {};
```

- Action nodes `a`, `b`, and `c` are defined and positioned in the diagram to represent the sequence of actions within the plan, as shown in Figure 3.

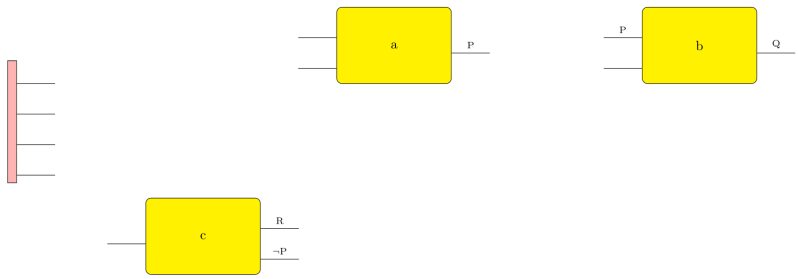

Figure 3: Action nodes a, b, and c representing the sequence of actions in the plan.

c) Goal State Node Added

Goal State Node

```
% Define the goal state node representing the final conditions to be achieved. This
    node is placed at coordinates (24, 0) in the diagram.
\node[GOAL, fill=red!30] (goal) at (24,0){};
```

- The goal state node (`GOAL`) is defined and positioned at (24,0) with a red fill, representing the desired final state of the POCL plan, as shown in Figure 4.

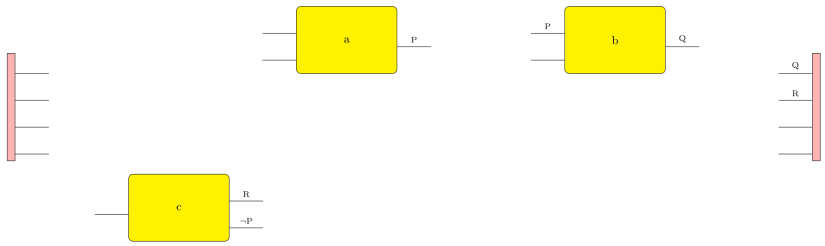

Figure 4: The goal state node representing the desired final state of the POCL plan.

d) Causal Links Drawn

- The causal links are drawn to connect the preconditions of one action to the effects of another, visually representing the dependencies between actions, as shown in Figure 5. This completes the sequence from the initial state to the goal state within the POCL plan.

Causal Links

```
    % Draw causal links between the actions to represent dependencies.
    \draw [causallink] (a-eff-1) node [dot]{} to [bend left] (b-pre-1) node [dot]{};
    \draw [causallink] (b-eff-1) node [dot]{} to [bend left] (goal-pre-1) node [dot]{};
    \draw [causallink] (c-eff-1) node [dot]{} to [bend right] (goal-pre-2) node [dot]{};
    \draw [causallink] (c) to [bend right, above] node {$<$} (a);
```

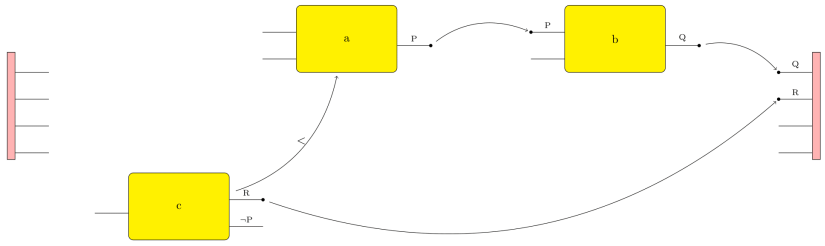

Figure 5: Causal links between actions representing dependencies within the POCL plan.

Example 2 (transportation.tex , This code belongs to yourplan.tex)

Introduction to Transportation Scenario

The transportation scenario is an example problem that involves moving objects from one location to another using a robot. The goal is to achieve a final state where all objects are in their designated locations by executing a series of predefined actions such as loading, unloading, and moving objects.

In this example, we will demonstrate how to use TikZ and custom styles to define a POCL (Partial Order Causal Link) plan diagram for a transportation scenario. This graphical representation shows the causal relationships between actions and how the system transitions from the initial state to the goal state.

Code

a) Initial State Node Added

Initial State Node

```
\begin{tikzpicture}
    % Define the initial state node representing the starting conditions.
    % This node is placed at the coordinates (0, 0) in the diagram.
    \node[INIT, fill=red!30] (init) at (0,0) {};
```

- The initial state node (INIT) is defined and positioned at the origin (0,0) with a red fill. This represents the starting conditions of the scenario, as shown in Figure 6.

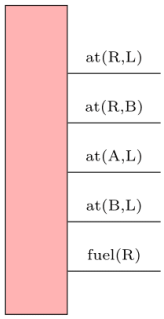

Figure 6: Initial state node representing the starting conditions of the scenario.

b) Load Action Nodes Added

Load Action Nodes

```
% Define the Load(A, L, R) action node, representing the robot R loading object A at
    location L. This node is placed at coordinates (7, 4.5) in the diagram.
\node[Load={A}{L}{R}] (LoadALR) at (7,4.5) {};

% Define the Load(B, L, R) action node, representing the robot R loading object B at
    location L. This node is placed at coordinates (7, 1.5) in the diagram.
\node[Load={B}{L}{R}] (LoadBLR) at (7,1.5) {};
```

- Two `Load` action nodes are added for the robot loading objects A and B at location L, as shown in Figure 7.

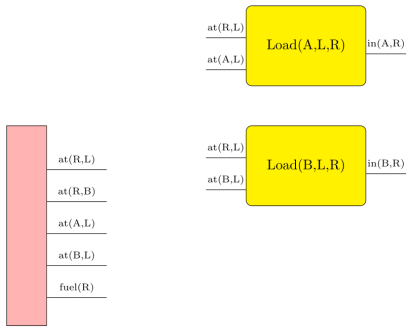

Figure 7: Load action nodes for the robot loading objects A and B at location L.

c) Move Action Nodes Added

Move Action Nodes

```
% Define the Move(L, P, R) action node, representing the robot R moving from location
    L to location P. This node is placed at coordinates (7, -1.5) in the diagram.
\node[Move={L}{P}{R}] (MoveLPR) at (7,-1.5) {};

% Define the Move(B, P, R) action node, representing the robot R moving from location
    B to location P. This node is placed at coordinates (7, -4.5) in the diagram.
\node[Move={B}{P}{R}] (MoveBPR) at (7,-4.5) {};
```

- The `Move` action nodes are added to represent the robot moving from location L to P and B to P, showing the transportation process in the diagram, as illustrated in Figure 8.

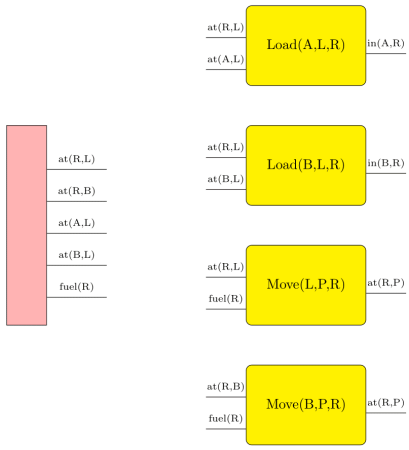

Figure 8: Move action nodes representing the robot moving from location L to P and from B to P, illustrating the transportation process.

d) Unload Action Nodes Added

Unload Action Nodes

```
% Define the Unload(A, P, R) action node, representing the robot R unloading object A at location P. This node is placed at coordinates (14, 3) in the diagram.
\node[Unload={A}{P}{R}] (UnloadAPR) at (14,3) {};

% Define the Unload(B, P, R) action node, representing the robot R unloading object B at location P. This node is placed at coordinates (14, -3) in the diagram.
\node[Unload={B}{P}{R}] (UnloadBPR) at (14,-3) {};
```

- The `Unload` action nodes are added, showing the robot unloading objects A and B at location P, completing the transportation process, as depicted in Figure 9.

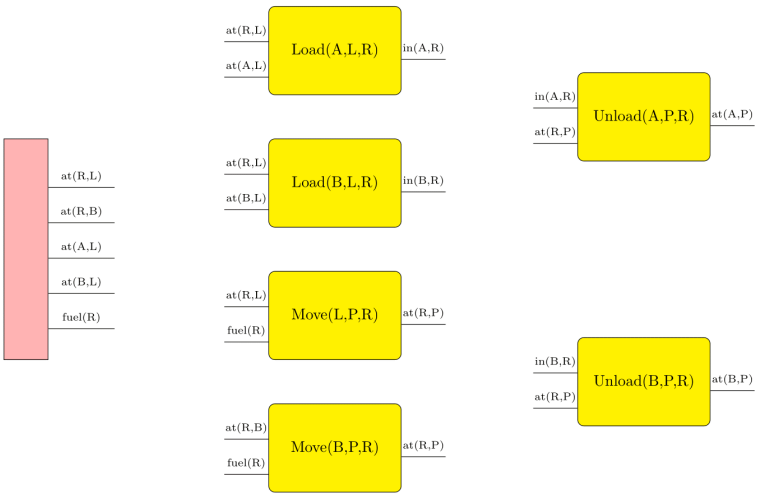

Figure 9: Unload action nodes showing the robot unloading objects A and B at location P, completing the transportation process.

e) Goal State Node Added

Goal State Node

```
% Define the goal state node representing the final conditions to be achieved. This
    node is placed at coordinates (21, 0) in the diagram.
\node[GOAL, fill=red!30] (goal) at (21,0) {};
```

- The goal state node (GOAL) is defined and positioned at (21,0) with a red fill, representing the desired final state of the scenario, as shown in Figure 10.

f) Causal Links Drawn

Causal Links

```
% Draw causal links between actions and states
\draw [causallink] (init-eff-1) node [dot]{} to (LoadALR-pre-1) node [dot]{};
\draw [causallink] (init-eff-3) node [dot]{} to (LoadALR-pre-2) node [dot]{};
\draw [causallink] (init-eff-1) node [dot]{} to (LoadBLR-pre-1) node [dot]{};
\draw [causallink] (init-eff-4) node [dot]{} to (LoadBLR-pre-2) node [dot]{};
\draw [causallink] (init-eff-1) node [dot]{} to (MoveLPR-pre-1) node [dot]{};
\draw [causallink] (init-eff-5) node [dot]{} to (MoveLPR-pre-2) node [dot]{};
\draw [causallink] (init-eff-2) node [dot]{} to (MoveBPR-pre-1) node [dot]{};
\draw [causallink] (init-eff-5) node [dot]{} to (MoveBPR-pre-2) node [dot]{};
\draw [causallink] (LoadALR-eff-1) node [dot]{} to (UnloadAPR-pre-1) node [dot]{};
\draw [causallink] (LoadBLR-eff-1) node [dot]{} to (UnloadBPR-pre-1) node [dot]{};
\draw [causallink] (MoveLPR-eff-1) node [dot]{} to (UnloadAPR-pre-2) node [dot]{};
```

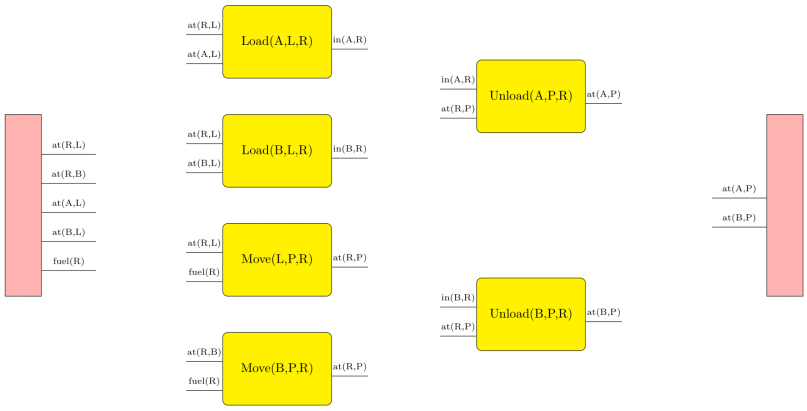

Figure 10: Goal state node representing the desired final state of the scenario.

```
    \draw [causallink] (MoveLPR-eff-1) node [dot]{} to (UnloadBPR-pre-2) node [dot]{};
    \draw [causallink] (MoveBPR-eff-1) node [dot]{} to (UnloadAPR-pre-2) node [dot]{};
    \draw [causallink] (MoveBPR-eff-1) node [dot]{} to (UnloadBPR-pre-2) node [dot]{};
    \draw [causallink] (UnloadAPR-eff-1) node [dot]{} to (goal-pre-1) node [dot]{};
    \draw [causallink] (UnloadBPR-eff-1) node [dot]{} to (goal-pre-2) node [dot]{};
```

- The causal links are drawn to connect the preconditions of one action to the effects of another, visually representing the dependencies between actions. This completes the sequence from the initial state to the goal state, as shown in Figure 11.

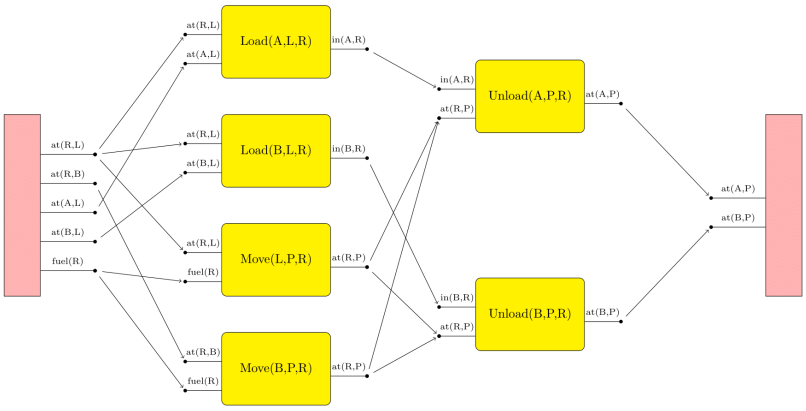

Figure 11: Causal links representing dependencies between actions, connecting preconditions and effects to complete the sequence from the initial state to the goal state.

The code provided step-by-step builds the final diagram, starting with the initial state (INIT), defining the actions (Load , Move , Unload), and finally connecting them through causal links to achieve the goal state (GOAL).

4 Compile Your Document

- Make sure you compile the main.tex file to generate the diagram. Ensure that all necessary files (domain.tex , yourplan.tex) are correctly referenced and present in the same directory.
- Use a LaTeX editor or command-line tools to compile main.tex . If successful, this will generate the desired diagram based on the TikZ instructions provided in the plan files.