Supplementary material for VizXP: A Visualization Framework for Conveying Explanations to Users in Model Reconciliation Problems

Paper ID: #97

Simplified Logistics

Original Domain

```
(define (domain logistics-simple)
  (:requirements :strips)
  (:predicates (is-package ?o) (is-truck ?t) (is-location ?l)
(is-airplane ?a) (is-hub ?hub) (in ?o1 ?o2))

(:action load-truck
  :parameters
   (?o ?t ?l)
  :precondition
   (and (is-package ?o) (is-truck ?t) (is-location ?l)
   (in ?t ?l) (in ?o ?l))
  :effect (and (not (in ?o ?l)) (in ?o ?t)))

(:action load-airplane
  :parameters
   (?o ?a ?l)
  :precondition
   (and (is-package ?o) (is-airplane ?a) (is-location ?l)
   (in ?o ?l) (in ?a ?l))
  :effect (and (not (in ?o ?l)) (in ?o ?a)))

(:action unload-truck
  :parameters
   (?o ?t ?l)
  :precondition
   (and (is-package ?o) (is-truck ?t) (is-location ?l)
        (in ?t ?l) (in ?o ?t))
  :effect (and (not (in ?o ?t)) (in ?o ?l)))

(:action unload-airplane
  :parameters
   (?o ?a ?l)
  :precondition
   (and (is-package ?o) (is-airplane ?a) (is-location ?l)
        (in ?o ?a) (in ?a ?l))
  :effect (and (not (in ?o ?a)) (in ?o ?l)))

(:action move-truck
  :parameters
   (?t ?l-from ?l-to ?l)
  :precondition
   (and (is-truck ?t) (is-location ?l-from) (is-location ?l-to)
   (is-location ?l) (in ?t ?l-from) (in ?l-from ?l) (in ?l-to ?l))
  :effect (and (not (in ?t ?l-from)) (in ?t ?l-to)))

(:action move-airplane
  :parameters
   (?a ?l-from ?l-to)
  :precondition
   (and (is-airplane ?a) (is-hub ?l-from) (is-hub ?l-to)
   (is-location ?l-from) (is-location ?l-to) (in ?a ?l-from))
  :effect (and (not (in ?a ?l-from)) (in ?a ?l-to))))
```

Original Problem

```
(define (problem problem-1)
(:domain logistics-simple)
(:objects airplane1 airplane2 location2 location3 location1 location4
city2 city1 truck2 truck1 package1)

(:init
(is-package package1)(is-truck truck1) (is-truck truck2)
(is-location city1) (is-location city2)
(is-location location1) (is-location location2)
(is-location location3) (is-location location4)
(is-hub location1)(is-hub location3)
(is-airplane airplane1) (is-airplane airplane2)
(in airplane1 location3) (in airplane2 location1)
(in truck1 location4) (in truck2 location1)
(in package1 location3)
(in location2 city1) (in location1 city1)
(in location3 city2) (in location4 city2))

(:goal (and (in package1 location2))))
```

Tweaked Domain (C1)

The preconditions for the `move-airplane` action were modified as follows

```
(:action move-airplane
  :parameters
   (?a ?l-from ?l-to)
  :precondition
%   (and (is-airplane ?a) (is-location ?l-from) (is-location ?l-to)
(in ?a ?l-from))
  :effect (and (not (in ?a ?l-from)) (in ?a ?l-to))))
```

Tweaked Problem (C2)

The location of `package1` was modified to be `location1` instead of `location3`

```
(define (problem tweaked)
(:domain logistics-tweaked)
(:objects airplane1 airplane2 location2 location3 location1 location4
city2 city1 truck2 truck1 package1)

(:init
(is-package package1)(is-truck truck1) (is-truck truck2)
(is-location city1) (is-location city2)
(is-location location1) (is-location location2)
(is-location location3) (is-location location4)
(is-hub location1)(is-hub location3)
(is-airplane airplane1) (is-airplane airplane2)
(in airplane1 location3) (in airplane2 location1)
(in truck1 location4) (in truck2 location1)
%  (in package1 location1)
(in location2 city1) (in location1 city1)
(in location3 city2) (in location4 city2))

(:goal (and (in package1 location2))))
```

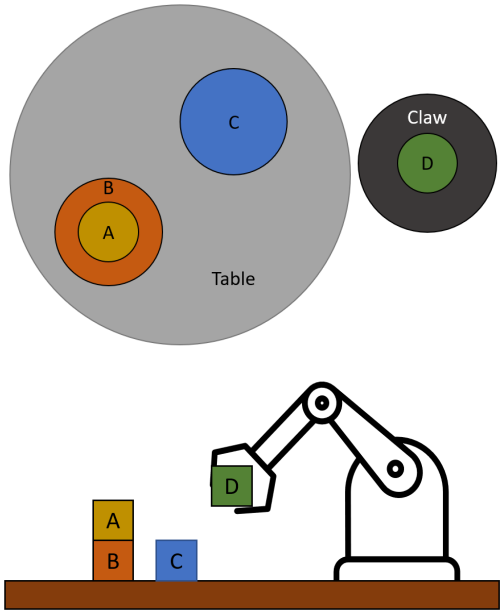

Figure 1: Top: Mock-up of a blocksworld instance using the container abstraction. Bottom: representation of the state

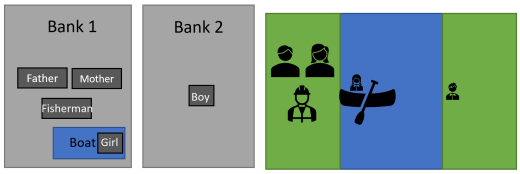

Figure 2: Two ways of visualizing the family and fisherman domain. Left: Using the general representation. Right: A tailor-made visualization using icons

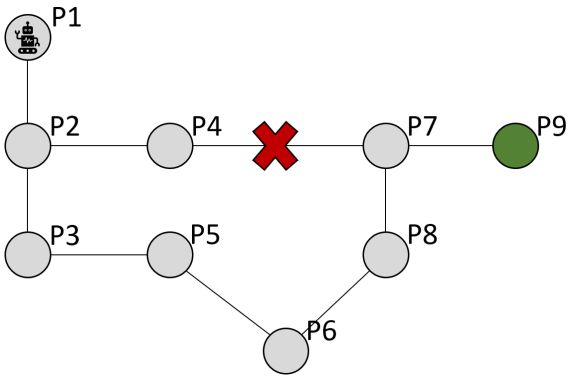

Figure 3: The Urban Search And Rescue (USAR) domain

Visualization

Container Abstraction: State-space

Here we provide some examples of the state-space visualization to illustrate its generalizability. Figure 1 shows a mock-up of how the BlocksWorld domain can be visualized using this abstraction technique, with the free blocks being available containers, in addition to the table and the claw. Free blocks can move to the claw if it is empty, and then to another free block or table. A similar approach can be used for domains like Tower of Hanoi, Parking (IPC 2011) and other similar ones.

The logistics example, as shown in the main text, can also be easily extended to domains like Mystery (IPC 1998), Depots (IPC 2002), and City Car (IPC 2014), among others. It is also possible to visualize domains like river-crossing with the family and fisherman, where specific design choices can be made to further tailor the state representation to the domain as shown in Figure 2. As an example, the size of the "boat" can be made such that it is obvious that two children can fit inside but only one adult will be able to. The visualizations can also be stylized by using colors and icons, but that will need to be done on a case-by-case basis.

Domains like VisitAll (IPC 2011), involving finding the shortest path that visits all nodes, can also be easily visualized, by utilizing a marker to denote which nodes have been visited, and edges to represent connections between nodes. In domains like Urban Search and Rescue (USAR), the abstraction can be used to represent a reduced version of the map for path-finding, as shown in Figure 3. It is possible to scale the connecting edges to represent the actual distances to overlay this on a map, but even without that information, it can provide a succinct overview of the various connections and blockages.

Limitations of the container abstraction All the examples described above are inherently positional, with the basic goal being to move things around. Thus, they lend themselves to the container abstraction readily. However, this may not always be the case. Specifically, for domains which rely on "properties" of objects to change, the actions do not necessarily move objects between containers. The Woodworking domain (IPC 2008), for example, requires modification of properties like size, surface finish and color, and almost all actions modify these. The container abstraction, thus, isn't well-suited for this domain on its own. However, it is possible to use augmentations like status icons and color to denote these properties as they change, but this will be tailored to every domain. The Rovers domain (IPC 2002) has actions that involve calibrating sensors and taking images. While sensor calibration can be shown as a property with status icons/color, it isn't obvious how image acquisition and communication can be visualized. It is possible to extend the container abstraction here to depict communication as a transfer of the file (in the rover) to the lander module via an edge, while taking a picture causes an icon to appear in the rover. All of these stretch the abstraction in certain ways, but it is possible to design visualizations using the abstraction as a base, especially when tailoring it to a certain domain.

If a certain property or action is hard to visualize, however, it is important to remember that the state-space visualization is only one part of VizXP. The action-space visualization is complete, and the actions' effects can be clearly observed by interacting with it, while such information can be hidden in the state-space visualization. Further, this in-

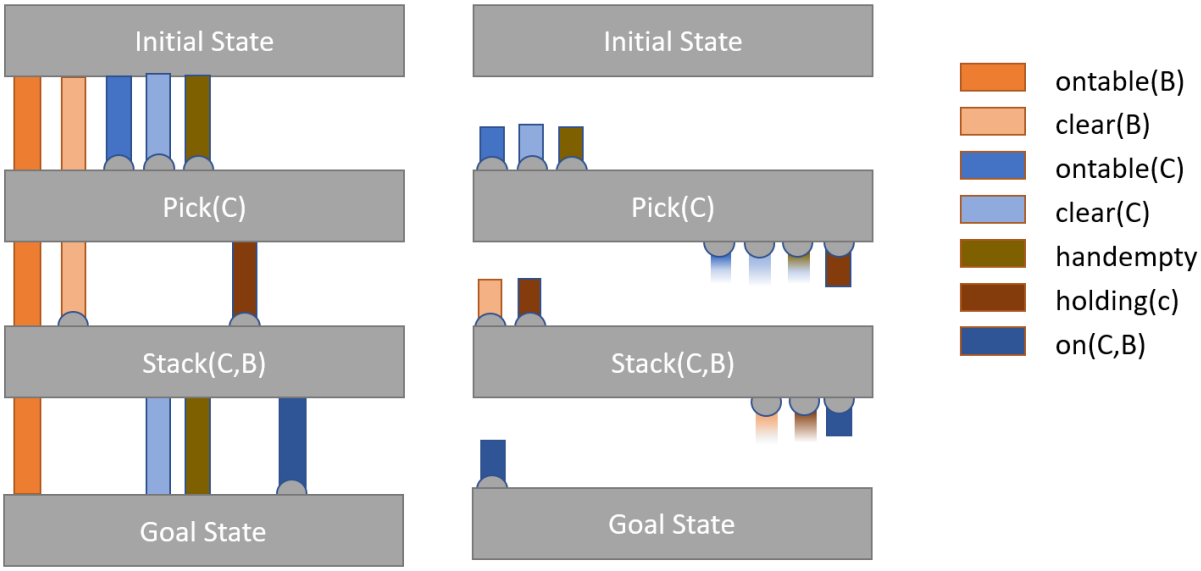

Figure 4: Left: A short plan visualized using Conductor. Right: The same plan, visualized using our approach

formation can be made available as a tooltip to the users for completeness of information (e.g., hovering over a wood block brings up its color, size and other properties).

Conductor vs Fact-flows

In this section we show the differences between Conductor's approach and ours. Figure 4 shows an illustrative example, solving a simple Blocksworld problem. The initial state consists of two blocks `B` and `C` on the table, and the goal is to stack `C` on `B`. This is achieved by simply picking up `C` and stacking it on `B`.

The primary difference between the two visualizations is that Conductor uses a dedicated column for each predicate, while our approach does not. This provides the benefit of being able to track predicate truth values across the actions for Conductor, but it also leads to a rapid growth in the number of unique columns as the domains grow complicated. In our case, we visualize just the preconditions for each action, and the effects, including the delete effects (fact flows that fade out), while omitting any predicates not relevant to the current action. The advantage of this approach can be seen by the fact that Conductor would need to be much wider as the number of blocks increased, needing 2 columns for each block (`ontable(x)` and `clear(x)`), but our visualization's width would remain the same. This avoids a lot of potential visual clutter.

To accommodate for the ability to track fact flows across actions, we propose the use of interactions: clicking or interacting with an action extends the associated fact flows across the entire plan, hiding fact flows for other actions. This lets users see the long term effects of an action, including information like which of the effects led to another action's execution later, or where the preconditions necessary for the selected action originated.

Also note that in the figure, the predicates are colored to distinguish them for illustration purposes, but in the actual system this need not be the case. The fact flows can all be the same color, with the actual predicates being made available on hover. This also allows us to use the highlight-based explanation technique proposed in the paper.

User Study: Comprehension Questions

Q1. *Given your plan and the explanation provided, why is your plan invalid? Please be as descriptive as possible.* (Open-ended)

Q2. *Do you feel you understand what information was omitted/conveyed incorrectly by Rob? If yes, what was that information?* (Open-ended)

Q3. *What are the corrections needed to your plan to make it achieve the goal, with the new information in mind?* (Open-ended)

Q4. *Where do you think the error in Rob's information was?* (Multiple choice)

i In the description of the actions and their preconditions.

ii In the description of the problem (start state or goal state).

Q5. *If applicable, identify areas with wrong or missing preconditions by clicking on the corresponding region. Double click to unselect.* (Users are shown a selection area where they can click on various actions.)

Q6. *If applicable, identify areas with wrong or missing start states by clicking on the corresponding region. Double click to unselect.* (Users are shown a selection area where they can click on various states.)

These questions ensured that the participants had to think about what the explanation meant, and hence allow us to see if they really understood it.

To evaluate the user responses, we scored them for each question, where the maximum score that can be achieved is 7 points. For the open-ended questions (Q1, Q2 and Q3), we manually read through the answers and assigned a correct and incorrect flag. For the rest of the questions, we had an answer key to check against the user responses. All questions except Q4 are worth 1 point. Q4 is worth 2 points if participants only select the correct answer, 1 point if they select both answers, and 0 otherwise.

User Study: System Feedback Questions (Likert Scale)

We asked the following questions as feedback from users who completed the study. The results are shown in Figure 5 for the people in the experimental group.

Q1. *I felt the visualization was useful in creating the plan.*
Q2. *The visualization helped me understand the execution of the plan.*
Q3. *I felt the visualization helped in understanding the difference in the plans.*
Q4. *I received the visualizations in a timely and efficient manner.*
Q5. *I found the various functions in this system well integrated.*
Q6. *I found the system easy to use.*

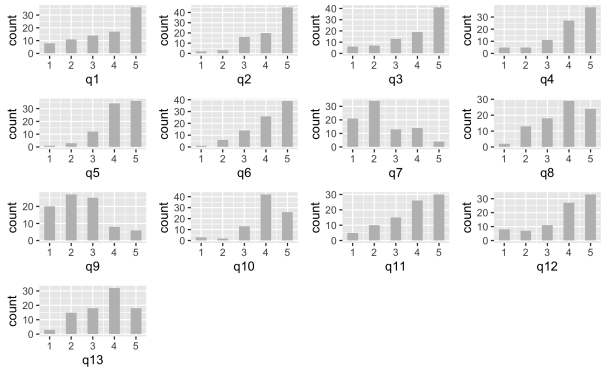

Figure 5: The statistics from the feedback questions. (0 = strongly disagree, 5 = strongly agree)

Q7. *I found the system unnecessarily complex.*
Q8. *I would imagine that most people would learn to use this system very quickly.*
Q9. *I needed to learn a lot of things before I could get going with this system.*
Q10. *I feel I understood the explanation presented by the system.*
Q11. *I feel the explanation presented helped me in identifying what was wrong in my original plan.*
Q12. *I felt it was easy to modify and correct my plan in the Plan Correction task.*
Q13. *I felt I needed the ability to test my plan in the Plan Correction task.*

Text: Editor and Visualization

Figure 6 shows the editor interface for users in the control group. The test functionality just indicates to them the steps which were wrong. We found no difference in the proportion of people who were able to complete Task 1 due to the Test visualization used, indicating that there was no bias introduced because of this. Figure 7 shows an example of the interface shown to users during Task 2(a).

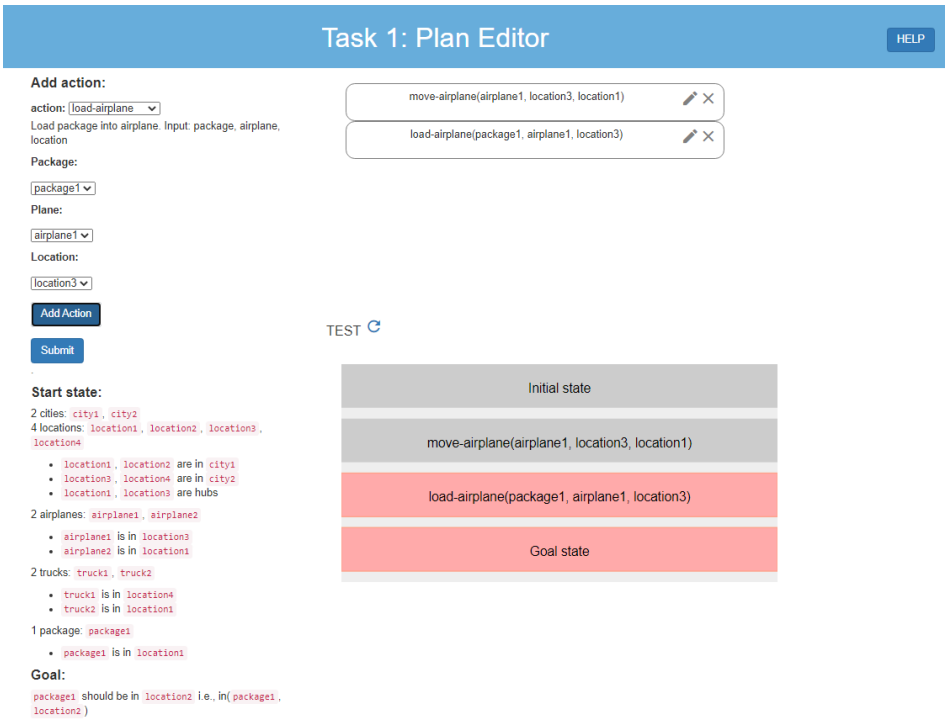

Figure 6: Editor interface for the control group

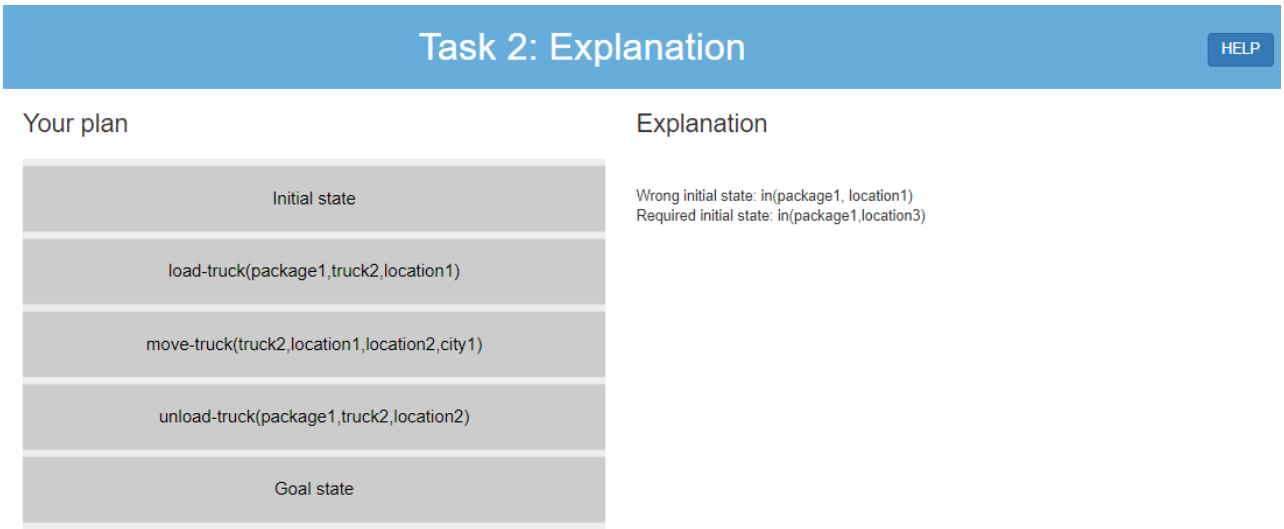

Figure 7: Explanation interface for the control group