G T r a n s Version 2.1:

User manual and reference

Chen Zhang, Michael T. Cox and Trivikram Immaneni

Department of Computer Science & Engineering
Wright State University

Technical Report WSU–CS–02-02

C O N T E N T S

The Structure of This Document:

The document is divided into four sections. The first section gives a brief introduction to the GTrans mixed-initiative planning system and the underlying PRODIGY/Agent planner. The User Interface section describes the various components of the Gtrans User Interface 2.0, including a brief description of each of the menus and menu items. This section has been designed to serve as a "Quick Reference" guide to GTrans. The Tutorial Section gives step-by-step instructions to run GTrans including explanations on how it interacts with PRODIGY, and it also gives a detailed description of how various operations can be performed in Gtrans. The final section is a list of references.

1 Introduction

1.1 Goal transformation

The world is under a continuous change and to plan in the real world, one adapts to these changes. Similarly, an AI planner has to adapt to world changes by refining the plan that is under construction. In addition to the changes made to the steps in the plan, the planner's goals may have to be changed as well -- this is the research topic addressed in *Goal transformations in continual planning* (Cox & Veloso, 1998; Cox, 2000). Some of the important issues that arise are:

1. Who should make these transformations – the human or the machine?
2. Is it better for some transformations to be made by humans and some by machines? If so, when should either take control?

Research is being done to address these issues. The PRODIGY (Carbonell et al., 1992; Veloso et al., 1995) planner demonstrates an important step towards goal transformations performed by a machine. During the planning process, PRODIGY keeps track of the changes in the real world through monitors (Veloso, Pollack, & Cox, 1998). They monitor the real world and inform the planner whenever there is a change. PRODIGY then transforms goals appropriately to adapt to the new state of the world. This worked well but had no means of evaluating its performance and effectiveness. To compare the goal transformations made by a machine to that made by a human, it was necessary to develop a system where a human can make these changes. GTrans (Cox, 2000; Cox, Kerkez, Srinivas, Edwin, & Archer, 2000; Cox & Zhang, 2003; Zhang, 2002) is such a system. GTrans works with the PRODIGY planner and allows a human to make goal transformations. This manual explains the working of the GTrans system.

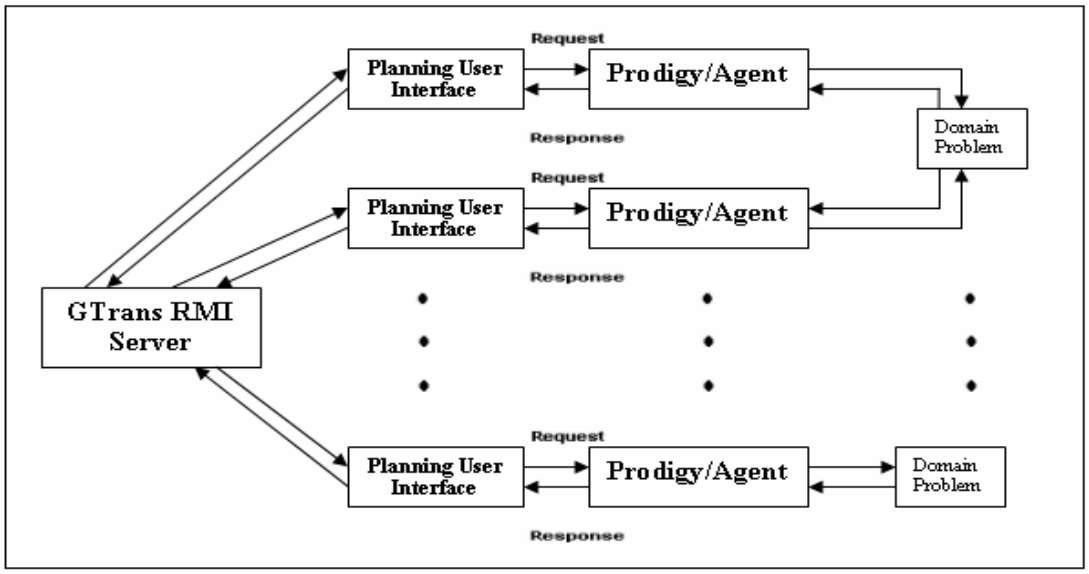


Figure 1 Basic structure of GTrans system.

1.2 What is GTrans?

Mixed-initiative planning combines human and machine problem solving processes so that the results are better than either can achieve alone. It is an interactive planning process in which, both humans and automated planners actively collaborate in the construction of plans. GTrans is mixed-initiative planning system that presents tasks to the user as a goal manipulation problems rather than a problem of search.

GTrans hides the planning algorithms and representations from the human planner while focusing on the goal manipulation process. It provides the user with a mechanism to directly manipulate the set of objects and initial state of the planning world, as well as the objectives of the planning problem. The user is able to define the objectives or goals and to assign particular resources to achieve these goals. When the underlying planner fails to generate a plan because of insufficient resources or because the planning world evolves, the user can asynchronously modify the goals and send them back to the planner for another round of planning, thus steering the planning process. The GUI is suitable mainly for spatial domains. The block diagram above (Figure 1) shows the basic structure of GTrans.

1.3 PRODIGY – A brief description.

PRODIGY is a nonlinear AI planner that searches for a sequence of actions that transform an initial state into a final state (goal state). The following sections include some of the basic terminology involved in problem solving with PRODIGY.

1.3.1 Domain

The domain specification constitutes of all those legal actions that can be performed by *operators* and legal inferences that can be made by *inference rules*. The operators and inference rules have a precondition expression that must be satisfied before they can be applied. The *effects list* summarizes how the application of the operator brings a change in that relevant *world*. An operator that has its parameters specified is called an *instantiated operator*.

The problem solving in PRODIGY is done by *means end analysis* in which a goal is picked that is not yet true and an attempt is made to find an operator to make it true. The preconditions of this operator become a goal and the process is repeated until all goals are attached to operators or are true.

1.3.2 Problem

The problem specification involves the *initial state* and the *goal state* that has to be achieved at the end of the plan.

The initial state consists of a list of objects and their types with a set of instantiated predicates. The goal specification is an expression that must be satisfied at any goal state.

PRODIGY 's decision cycle includes the following steps.

1. Given a list of pending goals, the system decides which goal to pursue next.

2. Given a goal, it decides which operator it will try to achieve the goal.

3. Given an operator, it decides what variable-bindings it will use to instantiate the operator.

4. Given instantiated operators with no open preconditions and a list of pending goals, it decides whether to apply an operator and hence change the current state or whether to choose a pending goal to solve.

When PRODIGY applies an instantiated operator in decision number four above, an action's effected are projected. Although this decision may be backtracked over, a final plan will consist of a sequence of these operator applications. For more information on PRODIGY refer to the PRODIGY manual.

2 The User Interface

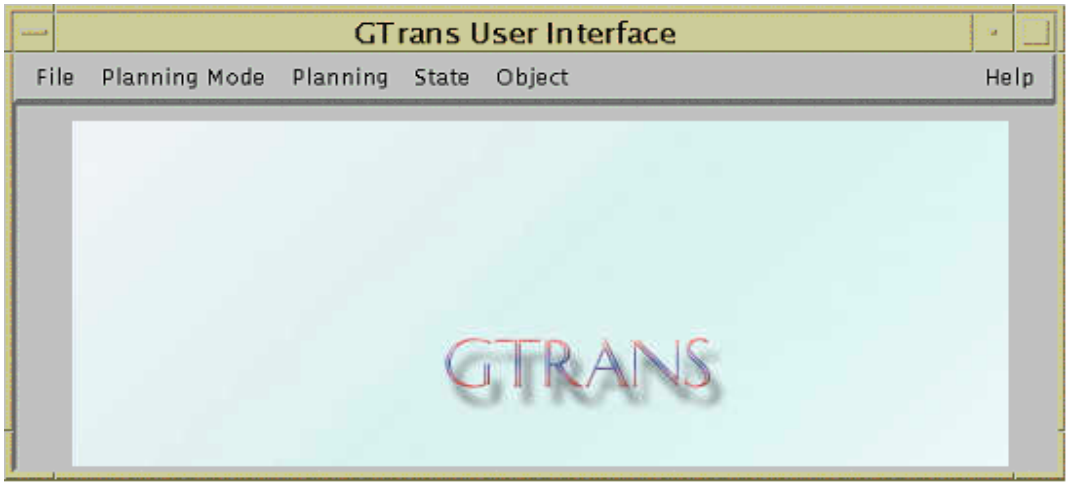


Figure 2 Initial GTrans User Interface

The GTrans User Interface (Figure2) consists of a main working area and various menus. The main working area (referred to as the canvas) is where the user can interact with objects. These interactions are through mouse drags and mouse clicks. Actions performed in the canvas are specific to the objects whereas menus are used to execute commands on the system as a whole.

2.1 Canvas

The canvas is used to place objects and interact with them. The user can place objects by selecting the desired object type from the menu (explained later) and then clicking on the canvas. The point where the mouse is clicked to place the object is recorded as the position of the object. An object is either mobile, in which case it can be moved around on the canvas, or stationary, in which case it occupies a fixed position. Movement of the mobile objects is carried out by mouse drags. When the mouse is moved over a mobile object, the cursor changes shape to a hand cursor indicating that the object can be dragged. The user may drag the mobile object to place it in a different position, or may drag it onto another object to set state or goal information. The user can also right-click on an object to set state or goal information.

2.2 Static Menus

There are various menus and menu items in the GTrans user interface. All of the menus are static and are present when the user interface is started, with the exception of two menus that are dynamically created for the domain on which the user prefers to work. The “File”, “Planning Mode”, “Planning”, “State”, and the “Object” menus are static menus, and remain present the entire time the user interface is running. These static menus are described below and the Dynamic menus are described in the next section.

2.2.1 File

This menu has two menu – items: “Load Domain” and “Exit” menu item. Figure 3 shows the File menu and its menu items.

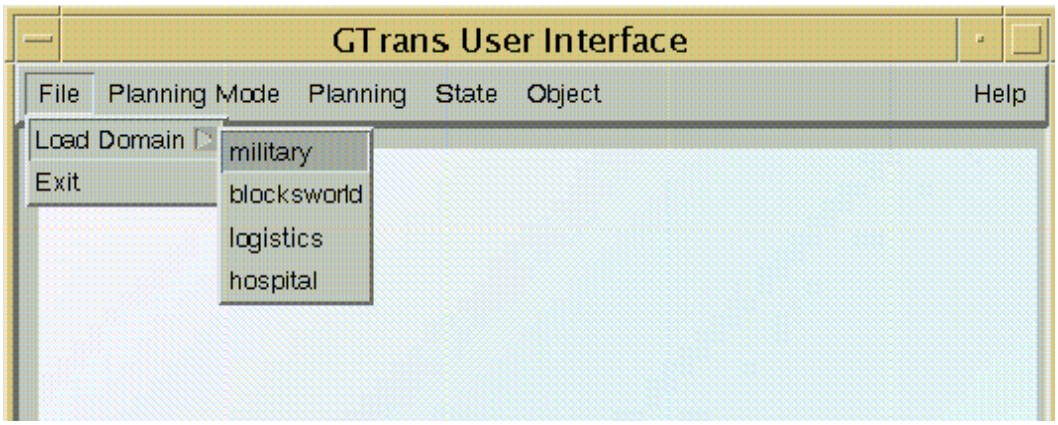


Figure 3 File Menu

2.2.1.1 File – Load Domain

This menu item is a sub-menu that lists all the PRODIGY domains on which a user can work. As of now only four domains of the many PRODIGY domains have been implemented to work with the Gtrans user interface: the "military", "blocksworld", "logistics" and "hospital" domains, though eventually all of the PRODIGY domains will be implemented to work with the GTrans user interface. When the user selects a domain from this list, dynamic menus are created for that domain and added to the menu-bar. The dynamic menus and their menu items are explained later.

2.2.1.2 File – Exit

This is the standard menu item found in almost every GUI. The user can exit from the Gtrans user interface and the system by selecting this menu item.

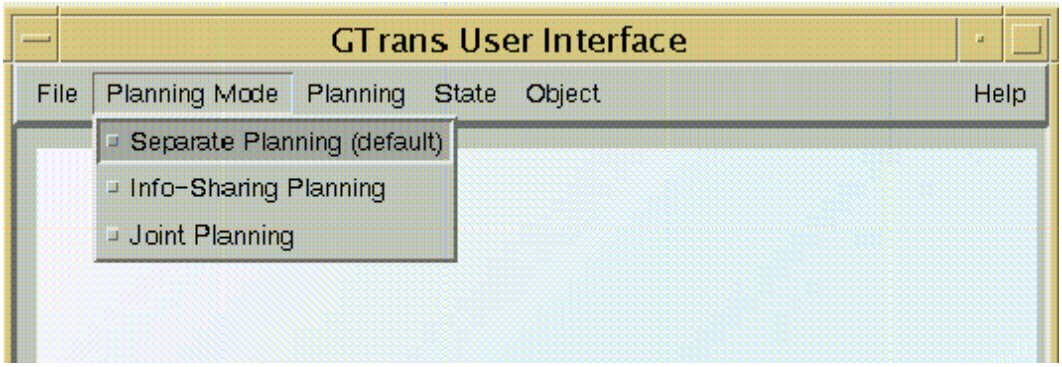


Figure 4 Planning Mode Menu

2.2.2 Planning Mode

GTrans is a multi-agent system in which several planners can jointly solve a problem. This menu (Figure 4) contains a radio menu item that allows the user to choose the planning mode in which he or she wants to work. Currently, there are three planning modes in which a user can work. If the user chooses to work in the "Separate Planning" mode, then he or she will perform planning separately without any access to the other planners' information. This is the default mode for all of the domains. In the "Info-sharing" mode, the user will be able to view the other planners' information. In the "Joint Planning" mode, multiple

planners can combine their resources to jointly solve a problem. For more information on the planning modes, please refer the tutorial.

2.2.3 Planning

This menu has commands associated with the planning problem being generated. The menu items in this menu are “List Goals”, “List Other Goals”, “Change Goals”, “Save Problem As”, and “Run”. Figure 5 shows the “Planning” menu and its menu items.

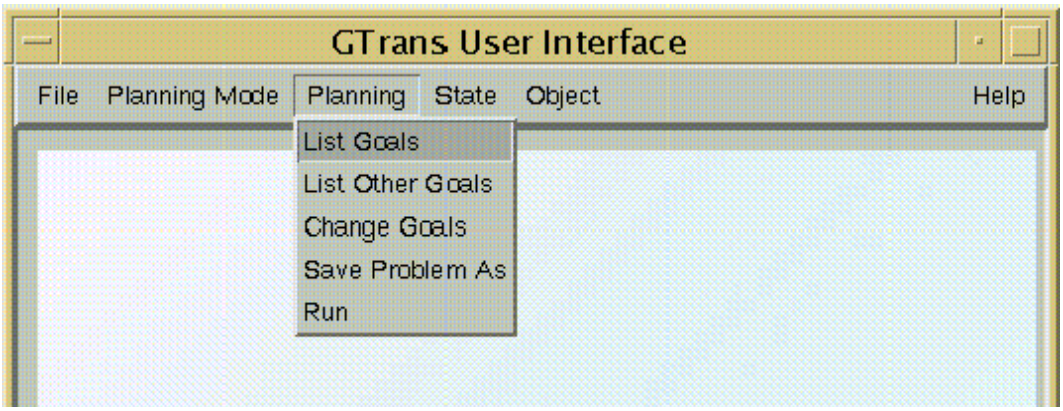


Figure 5 Planning Menu

2.2.3.1 Planning – List Goals

This menu item lists the goals set for the current problem. The goals are entered in a list box and this list box is then displayed in a new window. No operations can be performed on the listed goals. The list is for user information alone.

2.2.3.2 Planning – List Other Goals

This menu item is used only when a planner is working in the “Info-Sharing Planning” mode or in the “Joint Planning” mode. This menu item lists the goals of the problems on which the other planners are working. The goals are entered in a list box and the list box is then displayed in a new window. The list box also displays the identity of the planner whose goals are being listed. No operations can be performed on the listed goals. The list is for user information alone.

2.2.3.3 Planning – Change Goals

This menu item helps the user manipulate the goals set for the current problem. When the user chooses this menu item, the current goals are entered in a list box and this list box is then displayed in a new window. Goal transformations may be performed on the goals listed. To do so, a user has to select a goal that he/she wishes to transform and then click on either the “Delete” button or the “Change” button. If the user clicks on the “Delete” button, then the selected goal is removed from the problem. If the user clicks on the “Change” button, then a new “Perform Goal Change” window is displayed that contains the different transformations available. The user can use this window to perform a Goal type transformation, a Goal argument transformation, or a Negated goal transformation. To perform a transformation, the user has to select the required transformation from one of the drop-down menus and click on the “Select” button. Figure 6 shows a negated goal transformation being performed.

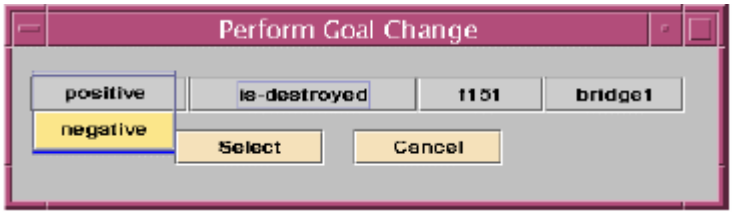


Figure 6 Negated Goal transformation

For more information on goal changes please refer to the Tutorial.

2.2.3.4 Planning – Save Problem As

This menu item creates a problem file from the information present in the scenario on which the user is currently working. This file is saved on disk so that the user can load and run it using the User Interface for PRODIGY (UIP). Often the execution of the plan and the steps taken while planning are important information. This information is hidden from the user who selects the menu item described next – "Run". In such cases the user will have to first save the problem to disk and then run it from the UIP.

2.2.3.5 Planning – Run

Once the scenario has been created, the use can use this menu item to send the problem to the PRODIGY planner for execution. When the user selects this menu item, a new "Perform Planning" window (figure 7) is displayed. This window helps the user to interactively define a plan. This window has a "Get Plan" button and a "Done" button. When the user clicks on the "Get Plan" button, GTrans sends the current problem to the PRODIGY planner for execution, and . PRODIGY will try to generate a plan. If PRODIGY succeeds in generating a plan, then the plan is displayed in the text area of the window. If PRODIGY cannot generate a plan, then PRODIGY returned is displayed in the text field. The user can then exit from the window by clicking on the "Done" button.

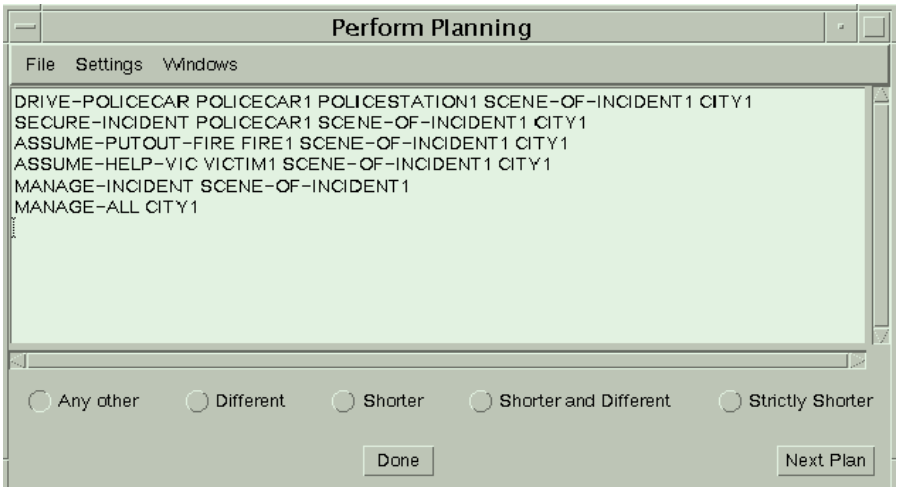


Figure 7 The Perform Planning Window

2.2.4 State

This menu has three menu items. One of them is a radio menu item with two choices, "Initial state" and "Final State". During the creation of the problem, the user has to set the initial states that the objects are in and later set the goal state for which PRODIGY has to plan. The items "Initial State" and

"Final State" help differentiate between the initial state and the goal state. The other two menu items are "List States" and "List Other States", described below. Figure 8 shows the "State" menu and its menu items.

2.2.4.1 State – Initial State

This is the default state that GTrans is in when started. In this state the user can set the initial states the objects are in. To do so the user has to right click on the object whose state is to be set, and a popup menu is displayed containing the various states that the objects can be in. The user has to select appropriate states for the objects. The creation of the popup menu is explained later.

2.2.4.2 State – Final State

The user has to enable this menu item in order to set the goal states of object(s). Once in this state, the user can set the goal states of as many objects as required. As in the case of "Initial State", the user has to right click in the object whose goal state is to be set. A popup menu is displayed containing the possible goal states that the object can be in, and the user can select the necessary goal state(s) from this menu. The user can go back and forth between the "Initial State" and the "Final State" any number of times.

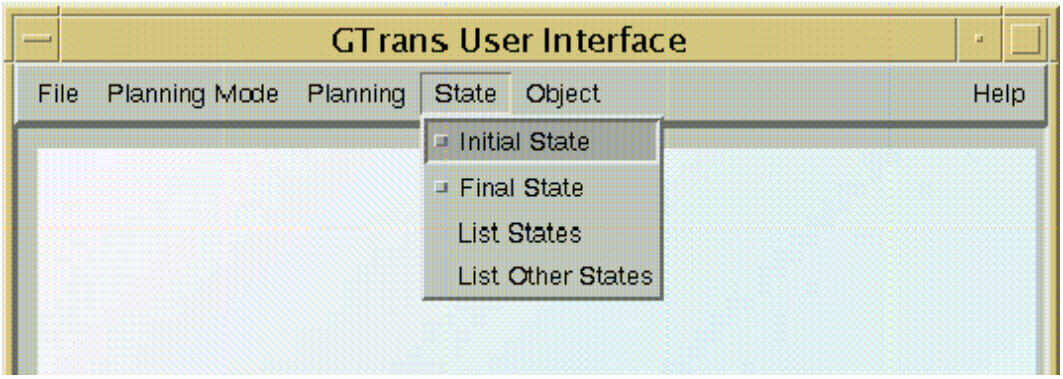


Figure 8 State Menu

2.2.4.3 State – List States

This menu item lists the states of the various objects in the current problem. The states are entered in a list box and this list box in displayed in a new window. No operations can be performed on the listed states. The list is for user information alone.

2.2.4.4 State – List Other States

This menu item is used only when a planner is working in the "Info-Sharing Planning" mode or in the "Joint Planning" mode. This menu item lists the (initial) states of the various objects in the problems on which the other planners are working. The states are entered in a list box and this list box in displayed in a new window. The list box also displays the identity of the planner whose object states are being listed. No operations can be performed on the listed states. The list is for user information alone.

2.2.5 Object

This menu has two static menu items and one dynamic menu item. The static menu items are "List Object" and "List Other Objects", and the dynamic menu item is "Add Object". The static menu items

are described here and the “Add Object” menu item is described in the Dynamic menus section. Figure 9 shows the “Object” menu.

2.2.5.1 Object – List Objects

This menu item lists the various objects present in the current scenario. The objects are entered in a list box and this list box in displayed in a new window. No operations can be performed on the listed objects. The list is for user information alone.

2.2.5.2 Object – List Other Objects

This menu item is used only when a planner is working in the “Info-Sharing Planning” mode or in the “Joint Planning” mode. This menu item lists the various objects in the problems on which the other planners are working. The objects are entered in a list box and this list box is displayed in a new window. The list box also displays the identity of the other planner whose objects are being listed. No operations can be performed on the listed objects. The list is for user information alone.

2.3 Dynamic Menus

The Dynamic menus, as the name suggests, are dynamically created at runtime. When a user loads a domain (or changes from one domain to another), these menus are created on the fly and added to the menu. Though these menus are specific to the domain on which the user chooses to work, their function is nevertheless identical in every domain. The dynamic menus are described in this section from the military domain’s perspective.

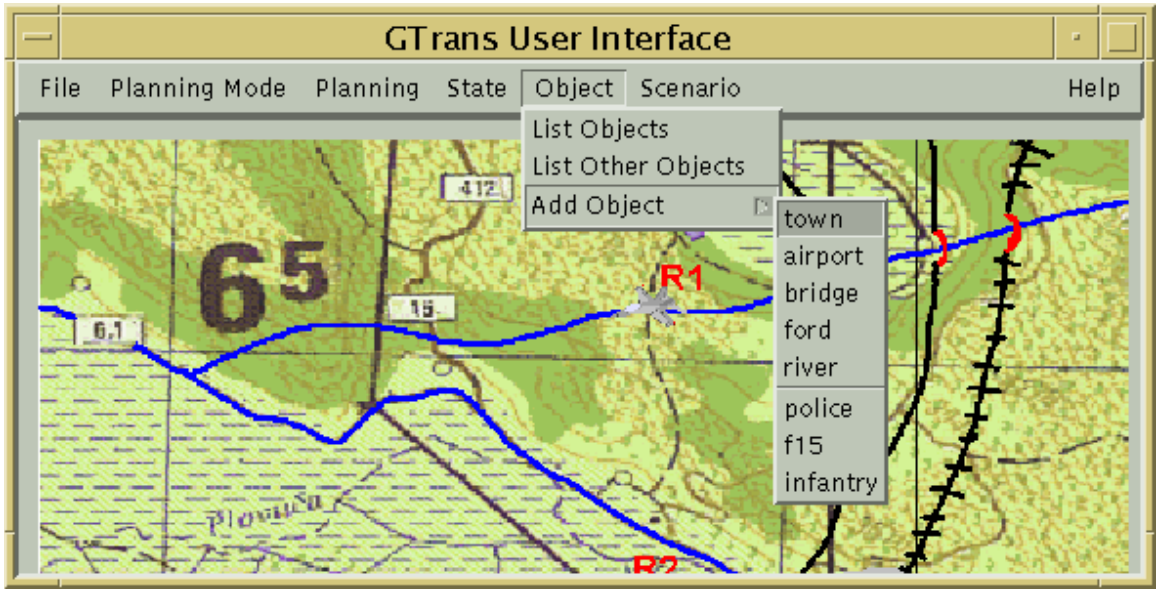


Figure 9 Object Menu

2.3.1 Object – Add Object

This is a dynamic menu item that is added to the static menu “Object” when the user loads a domain. It has a sub-menu containing the objects that can be created in the domain in which the user is working. Figure 8 shows the sub-menu in the “military” domain. As can be seen from the figure, the sub-

menu lists the objects that can be created in the "military" domain. When the user selects one of the objects (menu items) and clicks on the canvas, an instance of that object is created and placed at the position of the mouse click. As mentioned before, objects can be either stationary or mobile. In the sub-menu, the mobile objects are separated from the stationary objects by a thin line. In the above figure, “police”, “F15”, and “infantry” are mobile objects, and the rest are stationary. Counters are used to track the number of objects of a particular kind. Each time a new object is created, the current count is appended to it, giving it an identity. For example, the first time an "F15" object is created, it is appended with the number "1" making it "F15-1". The next "F15" object is called "F15-2" and so on. The identity appears as a label next to the object on the canvas. The “Add Object” sub-menu in the hospital domain is shown in figure 10.

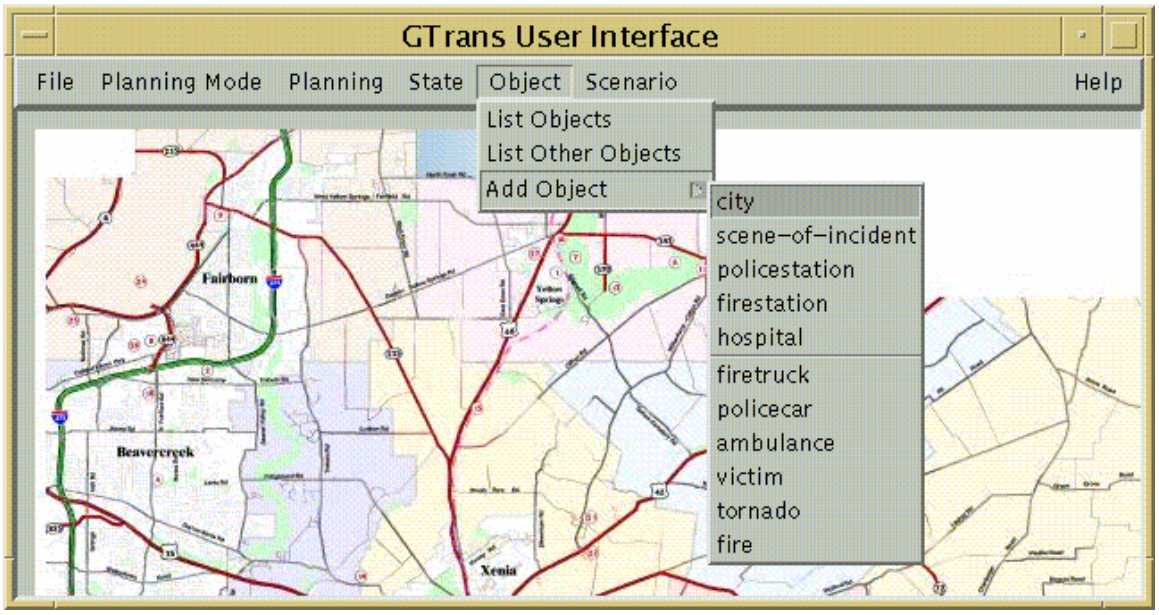


Figure 10 “Add Object” sub-menu in the Hospital domain

2.3.2 Scenario

In GTrans, the user can set up a scenario by placing objects on the canvas and setting their states. The setting up of a scenario may be a complicated and time-consuming process. Once the application is closed, the information is lost and the user will have to redo the scenario setup the next time. The “Save scenario” menu item provides the users with the facility to save a scenario. The saved scenario can then be loaded anytime using the “Load Scenario” menu item. In addition, this menu has the “Load Map” menu item that the user can use to load the background map. Figure 11 shows the “Scenario” menu.

2.3.2.1 Scenario - Load Map

This menu item is used to load a map onto the canvas. When this menu item is selected, a Load Window is displayed. The window lists all of the maps available for that particular domain. The user has to select a map from this listing and click on “OK”. The map is then loaded on the main-area. Figure 12 shows a map being loaded onto the main-area in the “military” domain. It also shows the “Load Map” window.

2.3.2.2 Scenario – Load Scenario

As mentioned earlier, GTrans allows the user to save the scenario information. To do so the user has to select the “Save Scenario” menu item (explained next). The “Load Scenario” menu item is used to load previously saved scenario information. When this menu item is selected, a “Load-Scenario” window is

displayed. This window lists all of the saved scenario files. The user has to select one of these and click "OK". Doing so will load the scenario as specified by the scenario file.

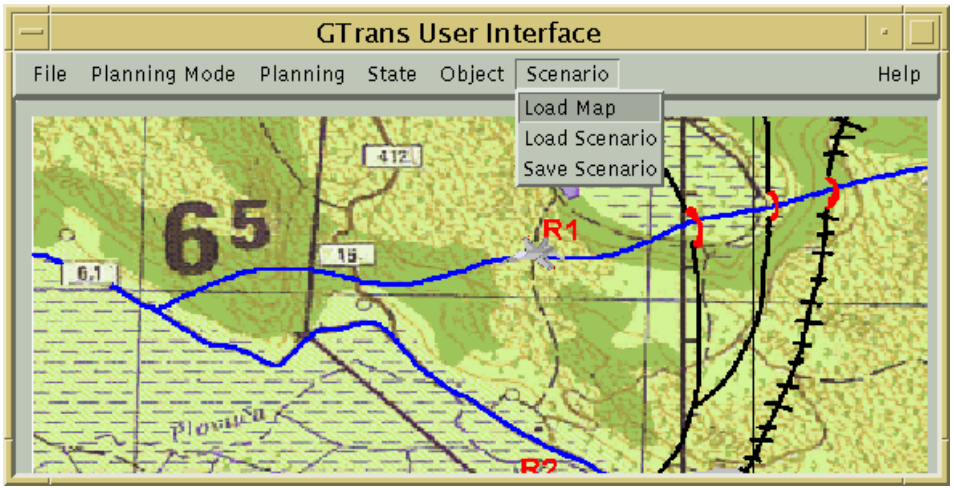


Figure 11 Scenario Menu

Figure 12 User Interface with a map and the "Load Map" window

2.3.2.3 Scenario – Save Scenario

This menu item is used to save the scenario information. Upon selecting this menu item, a "Save Scenario" window is displayed. This window lists the scenario files that have been saved previously. The users may either select one of these and overwrite it, or they may type in a new file-name. Once this is done and the "OK" button is clicked, the scenario information is saved to the file. Information that is saved in a scenario file is the object information, their position on the main-area, the states of the objects, and the goals.

3 Tutorial

This tutorial will walk the reader through the process of creating and running a problem on GTrans. For the sake of simplicity, the problem is run in "Separate planning" mode.
At present the setup instructions are specific to the cheops machine in COLAB2.

3.1 The Set Up

First Change the directory to "/usr/local/users/colab/gtrans2002/gtrans_multiple/gtrans"
>cd "/usr/local/users/colab/gtrans2002/gtrans_multiple/gtrans"

As explained earlier, in GTrans, several planners can jointly solve a problem. The planners exchange information via the GTrans RMI Server (see figure 1). GTrans RMI Server and Planning User Interface are implemented as a Java RMI application, in which Planning User Interface acts as a client whereas GTrans RMI Server plays the role of a server. Though only one server is available, there may be any number of clients communicating with the server. The server is responsible for keeping track of all of the clients. Whenever Planning User Interface is started, it registers with the running GTrans RMI Server and receives a unique number as its ID. The server then adds this client to its list of clients. When Planning User Interface exits, it informs the server so that the server can remove the client from the list.

The client sends every event that occurs in its User Interface to the server. The event could be the creation of an object, the setting of a goal or the state of an object, or the deletion of a goal. The server then broadcasts the event to all of the clients so that each client is aware of other clients' planning information.

3.1.1 The Server

The instructions to start the GTrans RMI Server are given here.

First, start the RMIregistry by executing the following command:
-rmiregistry &

An error message will appear at this point in case the registry is already running. Ignore the error message.

From the same directory, execute the following command:
- java newGtrans.GtransServerImpl &

This will start the RMI server and the server window will be seen. The server window is shown in Figure 13.

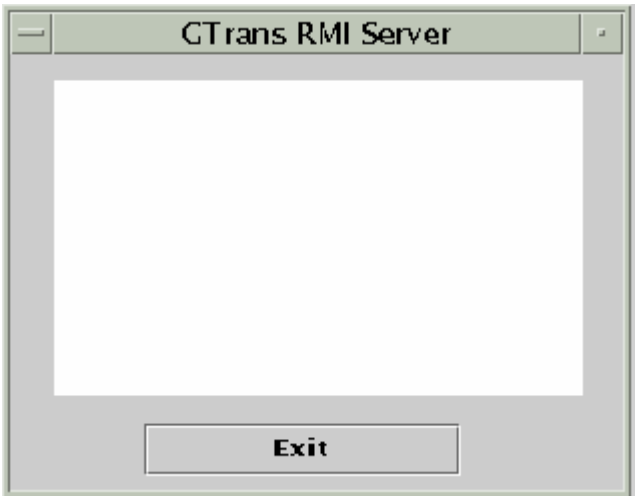


Figure 13 GTrans RMI Server

3.1.2 The Client

3.1.2.1 Running the registrar-loop (PRODIGY planner):

Open up a terminal.

Load emacs
- emacs &

Start Lisp
- C-x l
- Answer y to the following questions
  Load Prodigy 4.0?
  Load JADE?
  Load Prodigy?
  Load Monitor Code?

Load Wrapper?
- Answer n to any other (if others exist)

Load the loader file:
- (load "/usr/local/users/colab/gtrans2002/gtrans_multiple/loader")

Start the registrar loop:
- (registrar-loop 20000 t)

To start another registrar, repeat the process with a different port number (in the place of 20000). Note that each GTrans User Interface requires a separate registrar-loop (and a unique port).

3.1.2.2 Running the Gtrans User Interface

Start GTrans User Interface (GTrans 1): java newGtrans.GtransUserInterface 20000 &

The last argument (20000) is the port number and must be same as the one used to start the registrar-loop. At this point, the Gtrans User Interface (Figure 2) will appear.
Repeat this process to start more clients. Note that a separate registrar is needed for each client.

3.2 Loading a domain

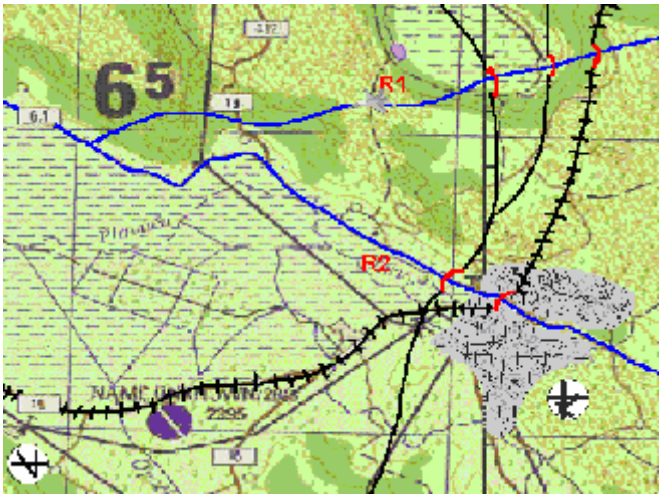
Now that the server and the user interface are running, the next step will be to load a domain. Theoretically, all of the domains available in PRODIGY can be implemented to work with GTrans. Currently, there are only four of these domains available on the planning user interface. For this tutorial the "military" domain is used. To load the "military" domain, select the File - Load Domain submenu and click on the "military" menu item. At this point, the default map will fill the canvas. This indicates that the "military" domain has been loaded. At present, the default map for the military domain is "aoi3.gif" (seen in Figure 14). The Dynamic menu "Scenario" will also appear on the main menu bar. If you want to work with any other domain, e.g., the "Hospital" domain, choose "Hospital" instead of "military" in the File – Load Domain sub-menu. When the domain is loaded, the client (User Interface) will register itself with the server. The server will assign a unique ID to the client and this ID is displayed in the server window. The server window at this point is shown in Figure 14.

Figure 14 GTrans RMI Server showing a client.

If you want to load another map, select the Scenario – Load Map menu item. This displays available maps in a File-Loader dialog box. For tutorial purposes the default map (aoi3.gif) is being used. This is shown in Figure 15.

The map has two rivers labeled "R1" and "R2". There are three bridges over "R1" and two over "R2". These bridges allow movement across the rivers and the goal we will be working towards will be to prevent movement across the rivers. To do so we will have to destroy each of the bridges. The assumption made is that to destroy one bridge we need one "F15" unit. The "F15" unit, as the name suggests, is a tactical-fighter air unit. Other information present on the map is the town and two airports. The town is in the right-bottom quadrant of the map. One airport is to the bottom-right of the town and the other, to the bottom-left of the map. The town is the gray area and the airports are the white circular regions with black lines.

The information on the map is visual and is not available for the GTrans system to use. So the next step will be to create objects that represent these entities and make them available to the GTrans system. By creating the objects and placing them at the appropriate positions, a mixed-initiative interface is created between the human and the machine, i.e. the visual information present on the map helps the user in setting up a scenario, and the objects the user creates to set up the scenario helps the GTrans system (machine) get information relevant to it.



R1 – river -1
R2 – river -2

Figure 15. Map file - aoi3.gif

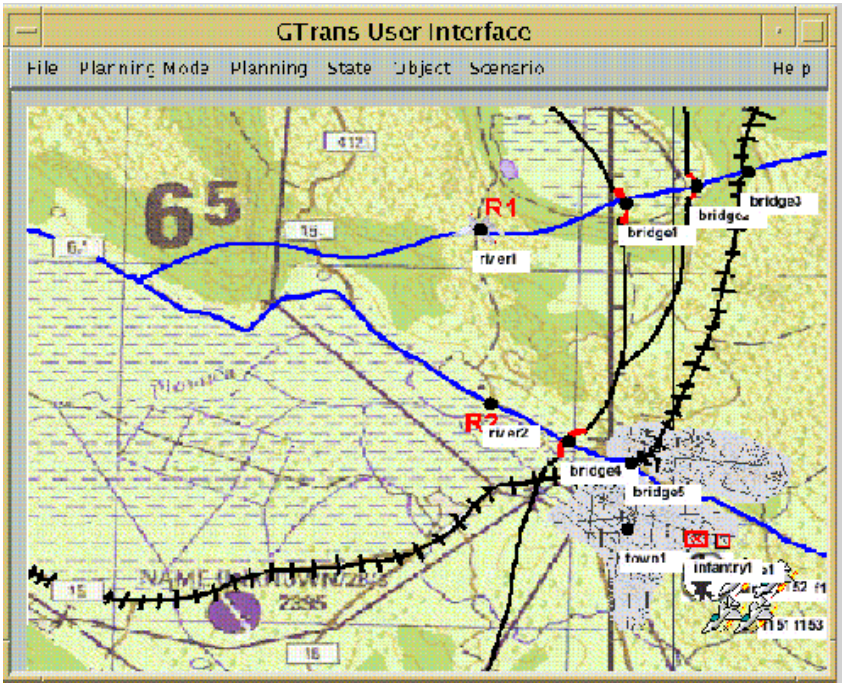


Figure 16 Tutorial Scenario

3.3 Creating Objects

Now that we are working in the “military” domain, the next step will be to create some objects to work on. As explained earlier, the map contained visual information about rivers, bridges towns etc. Now we have to create objects that represent them and provide information to the system. Each object can either be a mobile object (F15, Police, and Infantry) or a stationary object (Town, Bridge, and River). Stationary objects remain at the same position on the map over time, whereas mobile objects may change their

position. Mobile objects can also be viewed as resources that may be assigned to achieve goals. Selecting a particular type from the Object – Add Object sub-menu and then clicking on the map where it is to be placed creates these objects. Each object has a separate counter that keeps track of the number of objects created. Each time an object is created it is given a unique name (the current count value appended to the type of object) for its identity. For example, the first time the police unit is created it is given the name "Police-1". The next one police unit will be "Police-2", and so on. The Object's name appears as a label next to the object, as can be seen from figure 16.

For this tutorial we create five bridges, one town, two airports, one police, one infantry and four F15s. Place the rivers near the marks "R1" and "R2". Place three bridges across "R1" and two across "R2". Place one town and the two airports appropriately. Place the four F15s near airport-1 and the police and infantry near the town. The scenario should look similar to Figure 16.

Observe that the cursor changes to a hand cursor when moved over a mobile object. This indicates that the object can be dragged. The cursor remains the same when moved over a stationary object.

3.4 Setting States

Once the objects are created, their states should be set. To set the state, first make sure the system is in "initial state" mode. To check if the system is in "initial state", see if the "State – Initial State" radio menu item is enabled. If not, enable it by selecting it. Once in the "initial state" the states of the objects can be set till the state of the system is changed to "final state". To set the state of an object, right click on the object. A popup menu appears displaying all the possible states that the object can be in. Select the appropriate state that the object needs to be in. The process is shown in Figure 17.

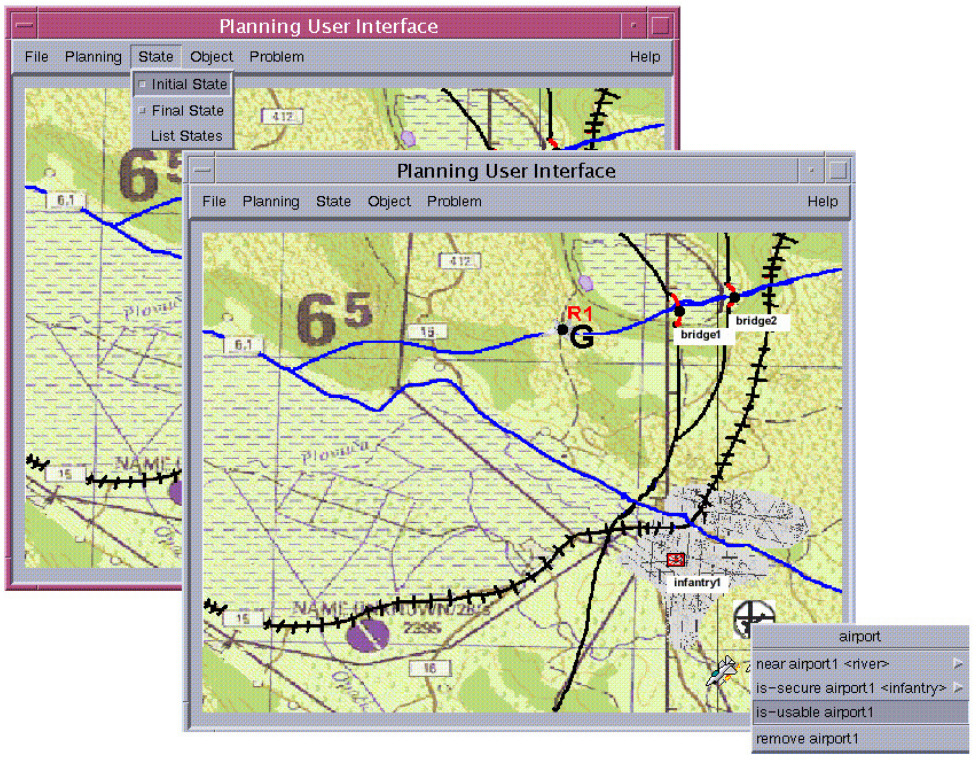


Figure 17 Setting initial state

The initial states of the various objects in the tutorial scenario are shown below.

Police-1 – is ready
Infantry-1 – is ready
F15-1 – is ready
F15-2 – is ready
F15-3 – is ready
F15-4 – is ready
Bridge-1 – enables-movement-over river-1*
Bridge-2 – enables-movement-over river-1*
Bridge-3 – enables-movement-over river-1*
Bridge-4 – enables-movement-over river-2*
Bridge-5 – enables-movement-over river-2*
Airport-1 – is-close-to river-2
Airport-2 – is-close-to river-2
Airport-1 – is-available
Airport-2 – is-available

* - This is assuming bridges one, two and three are placed over river-1 and the rest over river-2.

3.5 Setting the goals

Once the initial states are set, the goals for which PRODIGY has to plan must be set. There are two ways to set goals for the planning scenario. One way is similar to the way the initial state was set – the planner right clicks on the target object and selects the desired goals from a list of goals associated with the object (from the popup menu). This is shown in Figure 18.

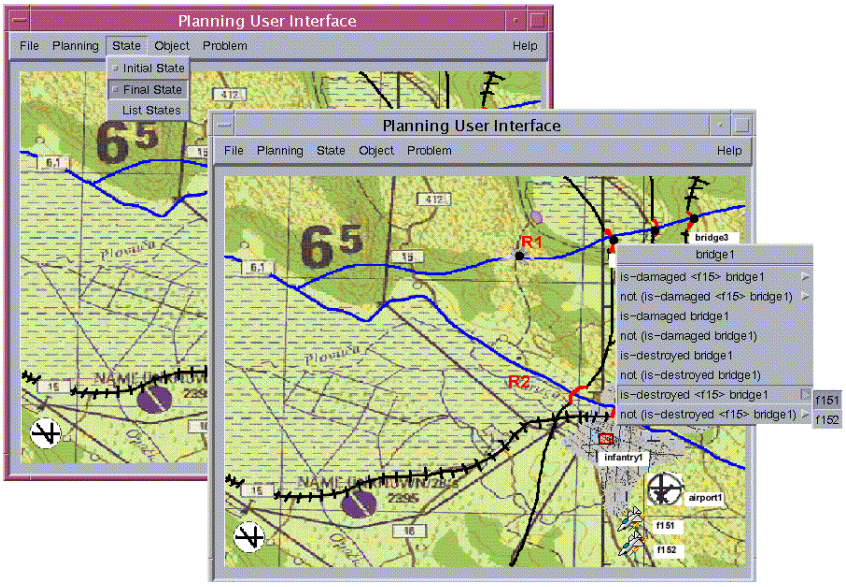


Figure 18 Setting goal state by right clicking the target object

The alternative is to define the goals by assigning resources. In order to assign a resource or a mobile object to a target object, the user drags the resource toward the target. Then he or she chooses the desired goal from the popup menu listing all the possible goals associated with the target that can be achieved using the resource (see Figure 19). In this way, the user defines the goal state and in the meanwhile assigns particular resources to achieve the goal state.

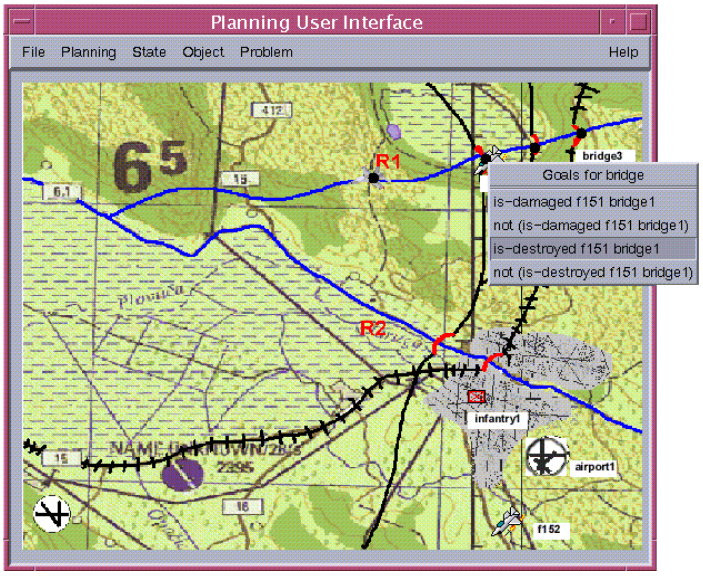


Figure 19 setting goal states by associating resources

In this tutorial, we'll make the two rivers impassable. So, there are two goals: make river-1 impassable, and make river-2 impassable. In order to set these goals, switch to final state and then right-click on river-1 object. A popup menu is displayed with the various goals that can be set. Select "outcome-impassable river-1" from the popup menu. You will see a "G" appear close to the river-1 object, indicating that a goal was set for river-1. Do the same for river-2. At this point, the User Interface would look like Figure 20.

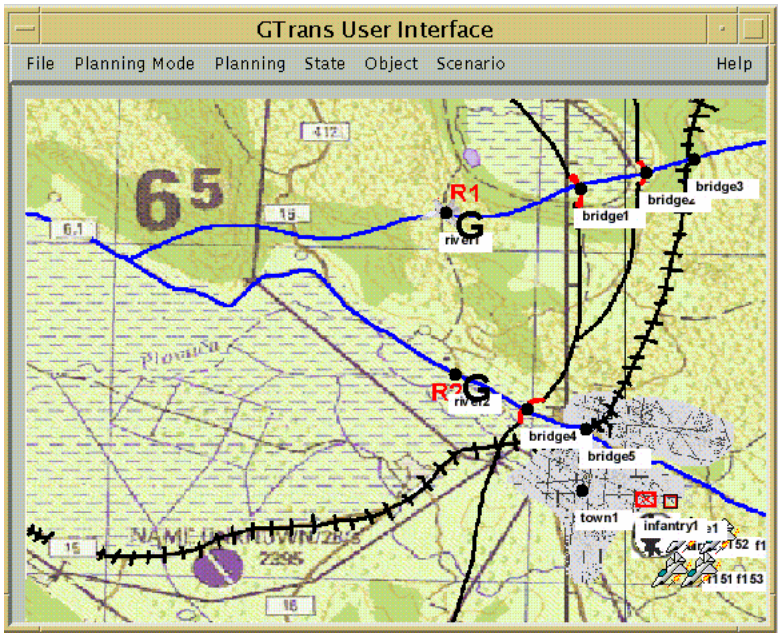


Figure 20 Tutorial Scenario after the Goals have been set

3.6 Running the Problem

Now that the scenario has been built, the next step will be to send the problem to PRODIGY for execution. At this point, you might want to save the scenario or the problem. This can be done by clicking on Scenario-Save Scenario or Planning-Save Problem respectively. Once this is done, we can click on Planning-Run to run the problem. At this point, a new window titled “Perform Planning” will appear which will help the user to interactively run the problem. This window has two buttons-“Get Plan” and “Done”. The user must click the “Get Plan” button to run the problem. This would cause the problem information to be sent to PRODIGY. The prodigy would then try to come up with a plan. The message returned by PRODIGY is then displayed in the text area in the window.

When the problem is run, you will notice that PRODIGY returns “No Plan” message. This is because there is no solution for this particular problem. The reason being that the domain information assumes that one F15 unit is required to destroy a bridge. The problem has four F15s and five bridges. So PRODIGY will not be able to come up with a plan. Such situations need a search in the space of goals and selection of a goal that can be reached. Such a search in the space of goals is called transformation of goals and is explained next.

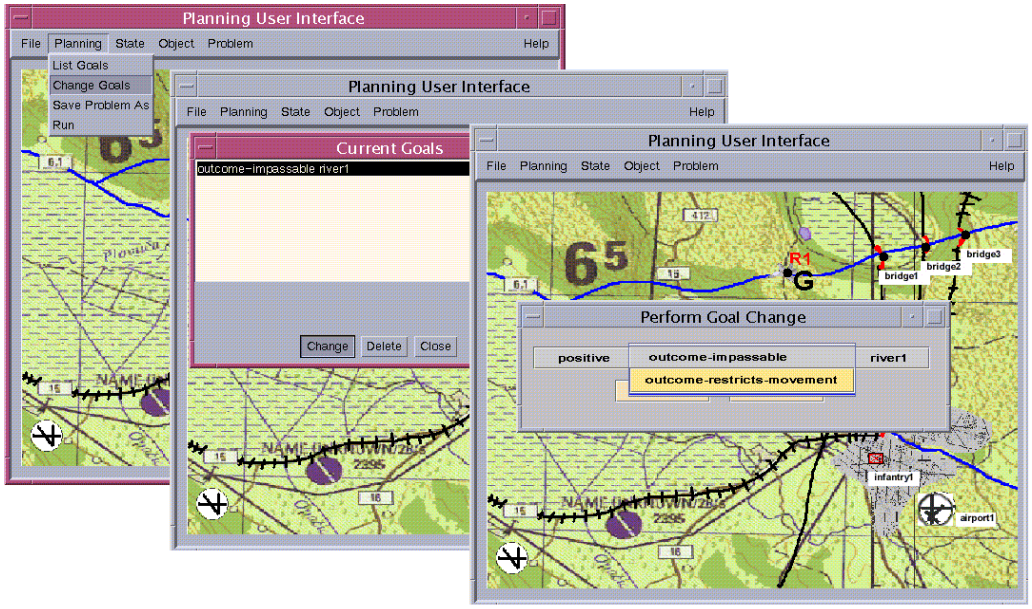


Figure 21 Transforming a goal

3.7 Transforming a Goal

As explained earlier transforming goals is an important step in planning. The present situation needs a transformation of one of the goals. Assuming the enemy region to be above river1 on the map, it is better if we make river-1impassable and restrict the movement over river-2 (just an assumption). So the goal of making river-2 impassable must be reduced to the goal of restricting movement across river2.

To do this transformation, select “Change Goals” from the “Planning” drop-menu. This will display a window titled “Current Goals” with a list containing the current goals (Figure-21). Select the goal “outcome-impassable River2” from this list and click the button labeled “Change”. A new window titled “Perform Goal Change” (Figure 21) is displayed containing a list of all possible transformations. In the

present case, we need to erode the goal to a lower expectation. This can be done by moving the goal predicate OUTCOME-IMPASSABLE to another goal predicate OUTCOME-RESTRICTS-MOVEMENT. This is done by choosing the latter from the drop-down menu and clicking the “Select” button. Notice that the transformation windows have a button labeled “Cancel”. This button can be used at anytime to abort the transformation process and destroy the transformation windows. Figure 21 shows the goal transformation process. Now that the goal has been transformed, run the problem again as explained earlier. You will notice that this time, a series of steps will be displayed in the text area of the “Perform Planning” window.

3.8 Planning Mode

The above problem was run in “Separate planning” mode for the sake of simplicity. The following section provides more information about the other planning modes.

Currently three planning modes - Separate planning, Information sharing planning, and Joint (collaborative) planning, have been implemented in GTrans.

As shown in Figure 22, when the planner chooses independent planning mode, he or she will perform planning separately without any access to the other planners’ information.

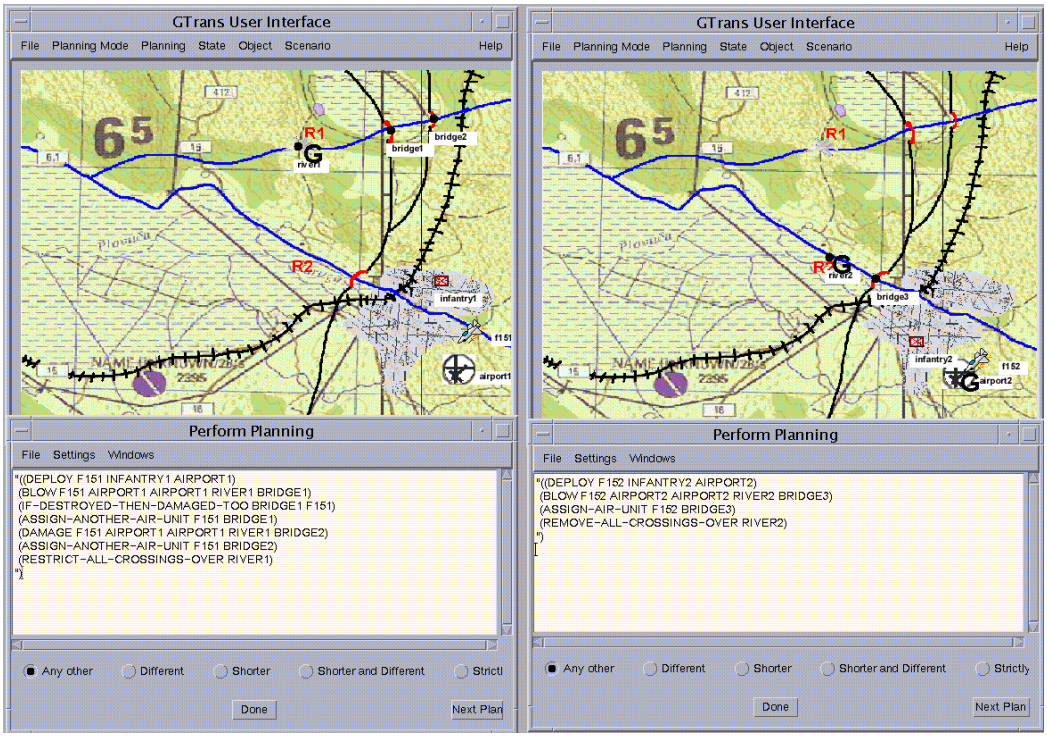


Figure 22 Planning under independent mode

However, once the planner switches to Information sharing mode, he or she will be able to view the other planners’ information while doing the planning (see Figure 23). The objects and the goals in the other planners’ scenario are drawn on the map and labeled in red. In this way, the planner is able to distinguish between his or her own information and the others’ information.

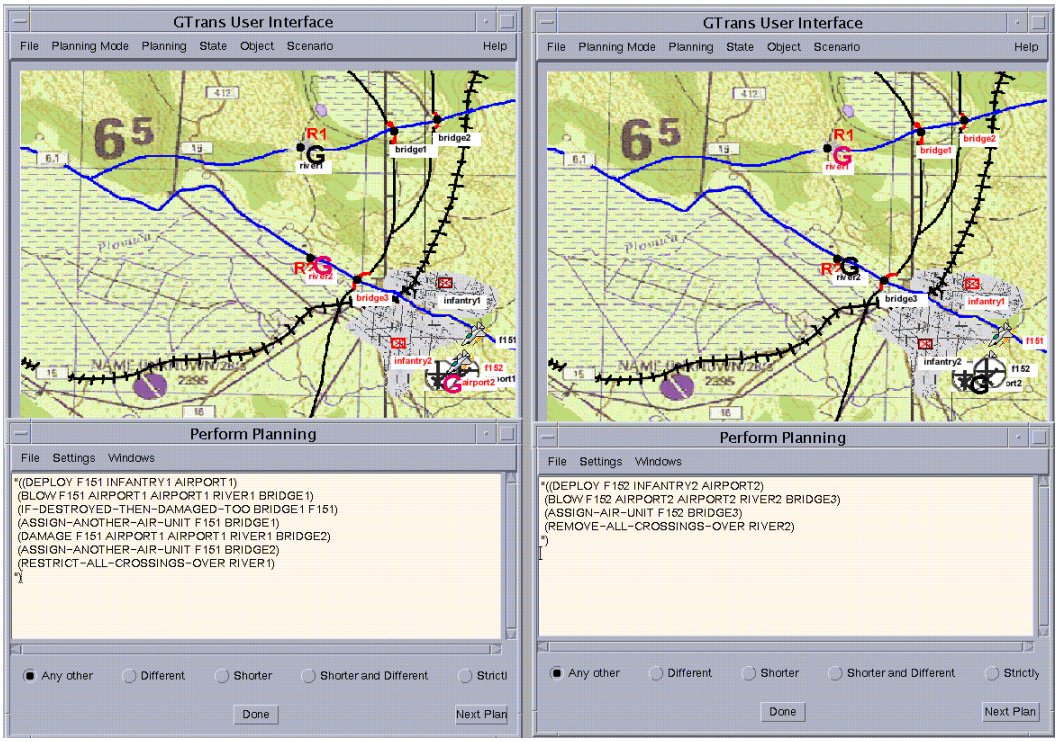


Figure 23 Planning under information sharing mode

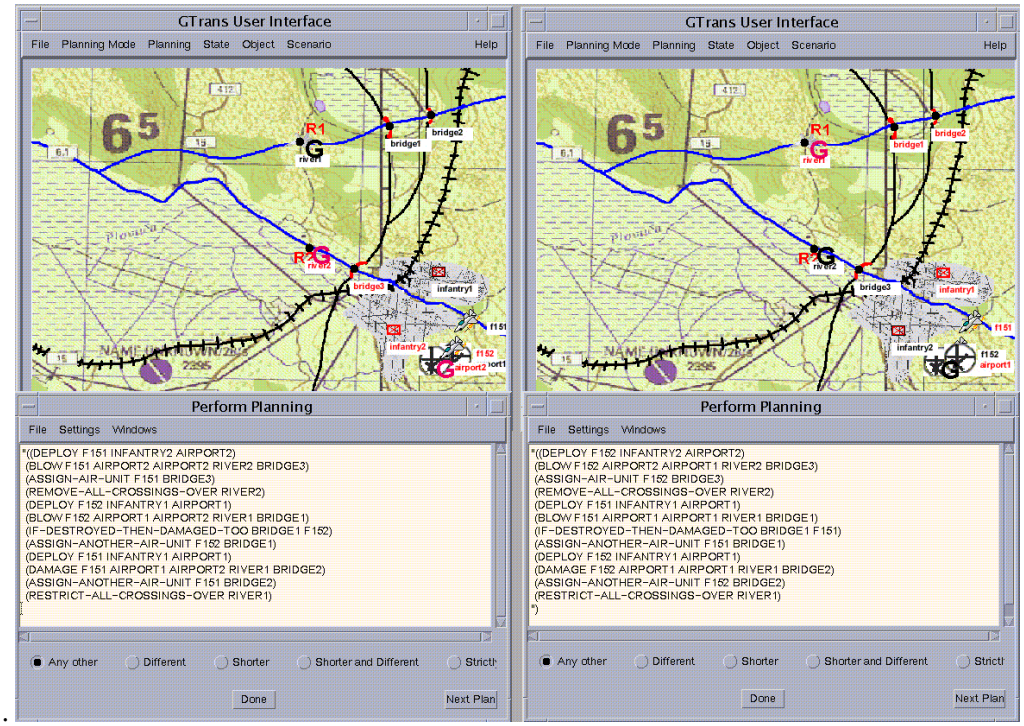


Figure 24 Planning under joint mode

Under joint planning mode, multiple planners actually perform collaborative planning, in which they have access to the partners' information and, furthermore, the individual information owned by each planner is combined into a more complex problem. In other words, each planner does part of the planning for a complex problem. Figure 24 shows an example of planning under joint mode. In the first Planning User Interface, the goal G1 is to restrict the movement across RIVER1. In the second Planning User Interface, the goal G2 is to make RIVER2 impassable. As can be seen, the two planners can visualize each other's information on the map. Besides this, when the planner sends a request to search for a plan to Prodigy/Agent, the problem information of each planner is concatenated into a larger problem and sent to Prodigy/Agent. Thus, the plan obtained by the individual planner achieves both G1 and G2.

4 References and Suggested Readings

1. Allen, J. (1994). Mixed-initiative planning: Position paper. [WWW document]. Presented at the ARPA/Rome Labs Planning Initiative Workshop. URL http://www.cs.rochester.edu/research/trains/mip/.

2. Blythe, J., Veloso, M. M., & de Souza, L. (1997). *The Prodigy user interface.* Technical Report, CMU-CS-97-114. Computer Science Department, Carnegie Mellon University.
0
3. Burstein, M. H., & McDermott, D. V. (1996). Issues in the development of human computer mixed-initiative planning. In B. Gorayska, & J.L. Mey (Eds.), *Cognitive Technology,* (pp. 285-303). Elsevier Science B.V.

4. Carbonell, J. G., Blythe, J., Etzioni, O., Gil, Y., Joseph, R., Kahn, D., Knoblock, C., Minton, S., Perez, A., Reilly, S., Veloso, M. M., & Wang, X. (1992). *Prodigy4.0: The Manual and Tutorial.* Technical Report, CMU-CS-92-150. Computer Science Department, Carnegie Mellon University.

5. Cohen, P. R. (1995). *Empirical methods for artificial intelligence*. Cambridge, MA: MIT Press.

6. Cox, M. T. (2000). A conflict of metaphors: Modeling the planning process. In *Proceedings of 2000 Summer Computer Simulation Conference* (pp. 666-671). San Diego: The Society for Computer Simulation International.

7. Cox, M. T., Edwin, G., Balasubramanian, K., & Elahi, M. (2001). Multiagent goal transformation and mixed-initiative planning using Prodigy/Agent. In N. Callaos, B. Sanchez, L. H. Encinas, & J. G. Busse (Eds.), *Proceedings of the 5th World Multiconference on Systemics, Cybernetics and Informatics, Vol. VII* (pp. 1-6). Orlando, FL: International Institute of Informatics and Systemics.

8. Cox, M., Kerkez, B., Srinivas, C., Edwin, G., Archer, W. (2000). Toward agent-based mixed-initiative interfaces. In H. R. Arabnia (Ed.), In *Proceedings of the 2000 International Conference on Artificial Intelligence, Vol. 1* (pp. 309-315). CSREA Press.

9. Cox, M. T., & Veloso, M. M. (1997a). Supporting combined human and machine planning: An interface for planning by analogical reasoning. In D. B. Leake & E. Plaza (Eds.), *Case-Based Reasoning Research and Development: Second International Conference on Case-Based Reasoning* (pp. 531-540). Berlin: Springer-Verlag.

10. Cox, M. T., & Veloso, M. M. (1997b). *Supporting Combined Human and Machine Planning: The Prodigy 4.0 User Interface Version 2.0* (Tech, Rep. No. CMU-CS-97- 174). Pittsburgh: Carnegie Mellon University, Computer Science Department.

11. Cox, M. T., & Veloso, M. M. (1998). Goal Transformations in Continuous Planning.

In M. desJardins (Ed.), *Proceedings of the 1998 AAAI Fall Symposium on Distributed Continual Planning* (pp. 23-30). Menlo Park, CA: AAAI Press / The MIT Press.

11.5 Cox, M. T., & Zhang, C. (2003). Planning as a mixed-initiative goal manipulation process. Manuscript submitted for publication.

12. Edwin, G. (2001). *Comas: coordination in multiagent systems***.** Master's thesis**,** Wright State University, Dayton, OH.

12.5 Edwin, G., & Cox, M. T. (2001). COMAS: CoOrdination in MultiAgent Systems. In N. Callaos, B. Sanchez, L. H. Encinas, & J. G. Busse (Eds.), *Proceedings of the 5th World Multiconference on Systemics, Cybernetics and Informatics, Vol. VII* (pp. 7-12). Orlando, FL: International Institute of Informatics and Systemics.

13. Ferguson, G., Allen, J. F., & Miller, B. (1996). TRAINS-95: Towards a mixed-initiative planning assistant. In *Proceedings of the Third International Conference on AI Planning Systems (AIPS-96),* Edinburgh, Scotland, May 29-31, 1996.

14. Ferguson, G., & Allen, J. F. (1998). TRIPS: An Integrated Intelligent Problem-Solving Assistant. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98),* Madison, WI, July 26--30, 1998.

15. Fikes, R. E., & Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2: 189-208.

16. Finin, T., McKay, D., & Fritzson, R. (1992). *An Overview of KQML: A Knowledge Query and Manipulation Language*. Computer Science Department, University of Maryland.

17. Freedman, R. (2000). Plan-Based Dialogue Management in a Physics Tutor. In *Proceedings of the Sixth Applied Natural Language Processing Conference, ANLP 2000.* Seattle.

18. Garay, M., Edala, N., Narayanan, S., & Eggleston R. (1999). An architecture for studying mixed-initiative planning and control in a complex, dynamic environment. In *Proceedings of the 1999 AAAI Workshop on Mixed-Initiative Intelligent Systems* (pp.115-118).

19. Hayes-Roth, B., & Hayes-Roth, F. (1979). A cognitive model of planning. *Cognitive Science*, 2, 275 – 310.

20. Mulvehill, A., & Cox, M. (1999). Using Mixed Initiative to Support Force Deployment and Execution. In M. T. Cox (Ed.), *Proceedings of the 1999 AAAI-99 Workshop on Mixed-Initiative Intelligence* (pp. 119-123). Menlo Park, CA: AAAI Press.

21. Narayanan, S., Bailey, W., Tendulkar, J., Wilson, K., Daley, R., & Pliske, D. (1999). Modeling real-world information seeking in a corporate environment. *International Journal of Human Factors and Ergonomics in Manufacturing*. 9(2), 1-31.

22. Oates, T. & Cohen, P. R. (1994). Toward a plan steering agent: Experiments with schedule maintenance. In *Proceedings of the Second International Conference on Planning Systems (AIPS-94) (pp. 134 –139).* Menlo Park, CA: AAAI Press.

23. Rich, C., & Sidner, C. L. (1998). COLLAGEN: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*. 8 (3/4).

24. Srinivas, C., Cox, M. T., & Laxminarayanan, V. (2000). *GTrans Version 1.0: User Manual and Reference*. Technical Report WSU-CS-00-02. Department of Computer Science and Engineering, Wright State University.

25. Traum, D. R., Allen, J. F., Ferguson, G., Heeman, P. A., Hwang, C. H., Kato, T., Martin, N., Poesio, M., & Schubert, L. K. (1994). Integrating Natural Language Understanding and Plan Reasoning in the TRAINS-93 Conversation System. In *Proceedings of the AAAI Spring Symposium on Active NLP,* (pp. 21-23).

26. Veloso, M. (1994). *Planning and learning by analogical reasoning*. Springer-Verlag.

27. Veloso, M. M., Carbonell, J. G., Perez, A., Borrajo, D., Fink, E., & Blythe, J. (1995). Integrating planning and learning: The PRODIGY architecture. *Journal of Theoretical* and Experimental Artificial Intelligence, 7(1), 81-120.

28. Veloso, M. M., Pollack, M. E., & Cox, M. T. (1998). Rationale-based monitoring for continuous planning in dynamic environments . In R. Simmons, M. Veloso, & S. Smith (Eds.), *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems* (pp. 171-179). Menlo Park, CA: AAAI Press.

29. Zhang, C. (2002). *Cognitive Models For Mixed-Initiative Planning*. Master's thesis**,** Wright State University, Dayton, OH.