Mixed-Initiative Planning: Supporting combined human and machine decisions

Michael T. Cox

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213-3891

http://www.cs.cmu.edu/~mcox

Planning is NP Hard

Approaches to Intractability:

- Use heuristic search methods
- Domain engineering
- Ignore the problem
- Leverage human experience.

Mixed-Initiative Planning:

Humans *and* machines are both intimately involved in the planning process.

Goal of Mixed-Initiative Planning:

To more fully incorporate the user into the planning process

Intended Result:

Better quality plans than either machines or humans can generate alone

Efficient Planning

Challenges of Mixed-Initiative Planning

- Human Planning Rationale Unavailable
- Idiosyncratic Planners
- Representation Discrepancies
- Concurrent Planning and Execution
- Ambiguous Domain Theory
- Adversarial Agents
- Uncertainty

Three Thrusts for Mixed-Initiative Planning

1. Create a more effective user interface to planning systems.
2. Allow users greater flexibility in specifying planning goals.
3. Allow user to dynamically change goals during planning.

Overview

Introduction

Traditional AI Planning

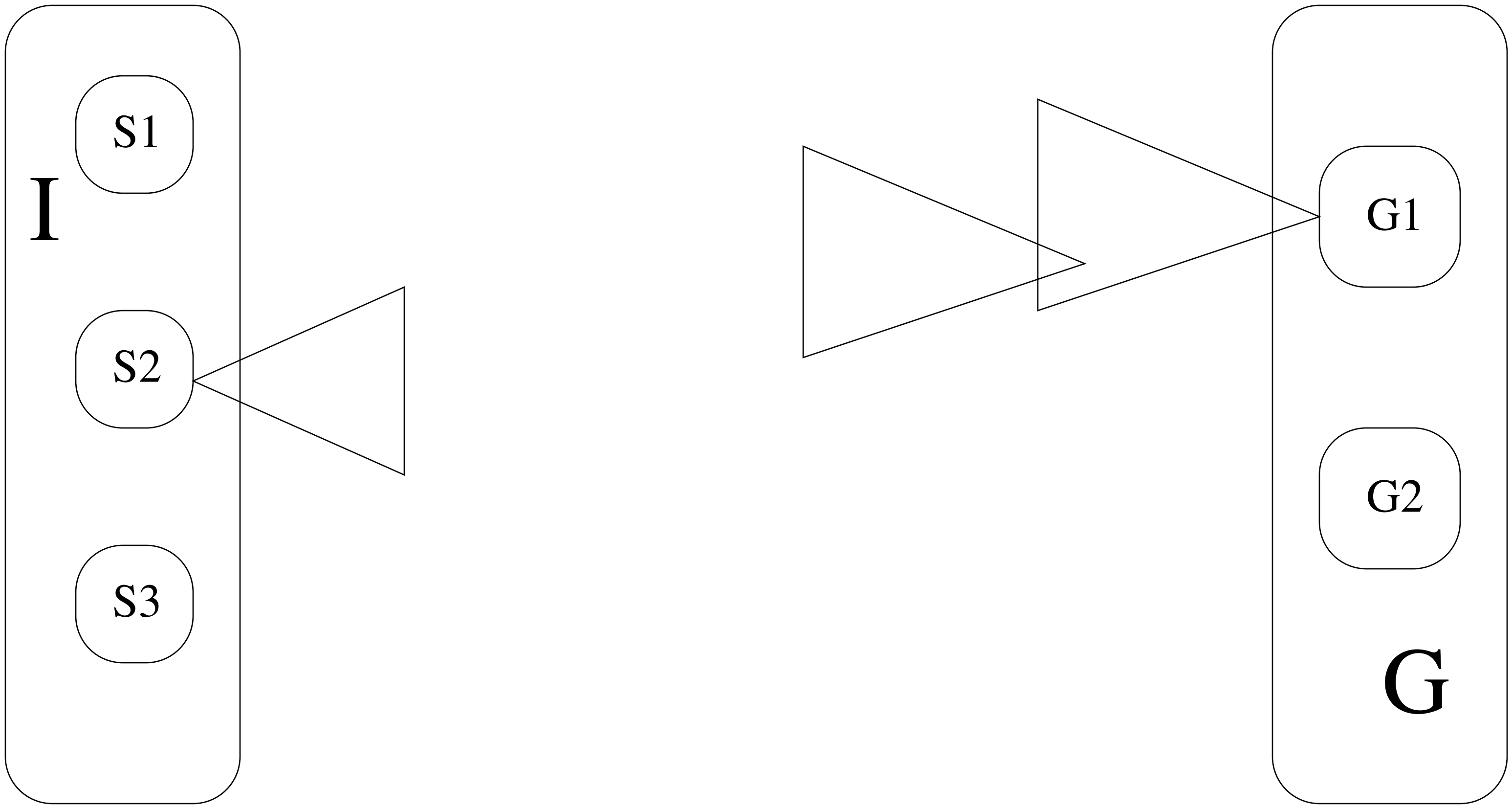
Mixed-Initiative Planning

- The PRODIGY Graphical User Interface
- Mixed-Initiative Case-Based Planning
- Continuous Mixed-Initiative Planning and Execution

Future Research

Conclusion

Planning as Search

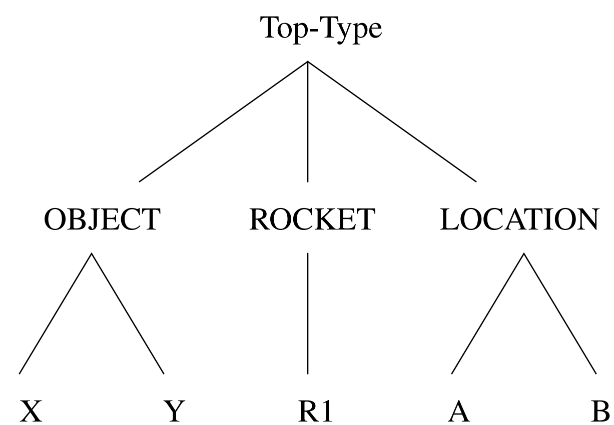


The One-Way Rocket Domain

Load-Rocket (<obj> <place> <rocket>)
<obj>: type OBJECT
<place>: type LOCATION
<rocket>: type ROCKET
Pre: (at <obj> <place>)
(at <rocket> <place>)
Del:(at <obj> <place>)
Add:(inside <obj> <rocket>)

Unload-Rocket (<obj> <place> <rocket>)
<obj>: type OBJECT
<place>: type LOCATION
<rocket>: type ROCKET
Pre: (inside <obj> <rocket>)
(at <rocket> <place>)
Del:(inside <obj> <rocket>)
Add:(at <obj> <place>)

Move-Rocket (<rocket>)
<rocket>: type ROCKET
Pre: (at <rocket> LocA)
Del:(at <rocket> LocA)
Add:(at <rocket> LocB)



Initial State

Goal State

State Loops

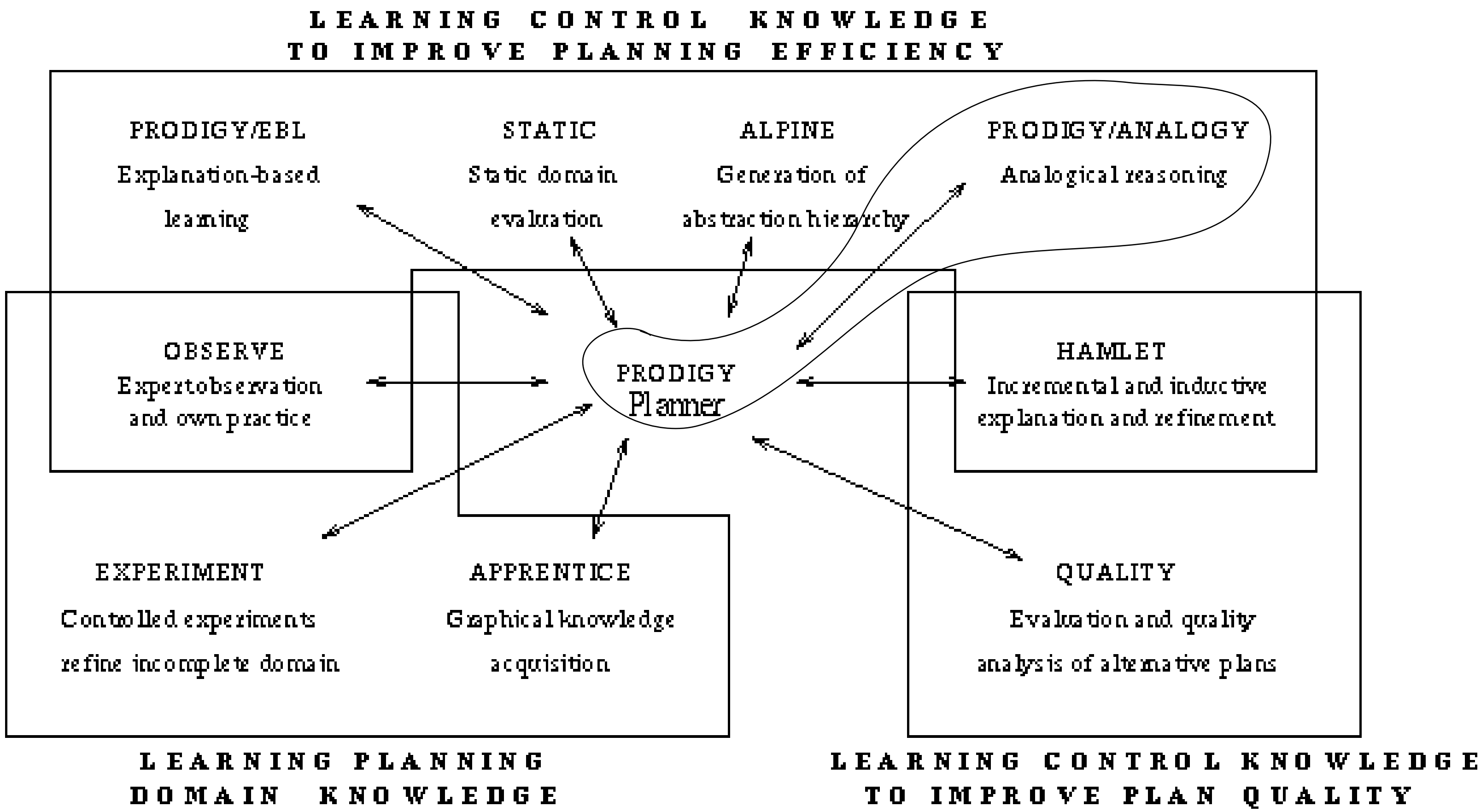
(Can load X, then unload X, indefinitely)

Non-independent Goals

(Cannot solve one goal, then solve the other)

The PRODIGY Planning and Learning Architecture

http://www.cs.cmu.edu/~prodigy



Top-level view of the planning algorithm (Prodigy 4.0)

1. Initialize.
2. Terminate if the goal statement has been satisfied.
3. Determine which goals are pending
4. Determine if there are any operators with satisfied preconditions that could be applied to the state as the next plan step.
5. Choose to subgoal on a goal or to apply an operator:
   * To subgoal, go to step 6 * To apply, go to step 7
6. Select a pending goal, an instantiated operator that can achieve it; go to step 3.
7. Change the state according to an applicable operator; go to step 2.

The PRODIGY Graphical User Interface:

Access to the *planning process*

Inserting human users in the planning loop presents three applied problems:

1. Styles differ in human and machine planning.

2. Visualization of the plan and planning process is complex, but necessary.

3. Humans range in experience with respect to the planning domain and the planning technology.

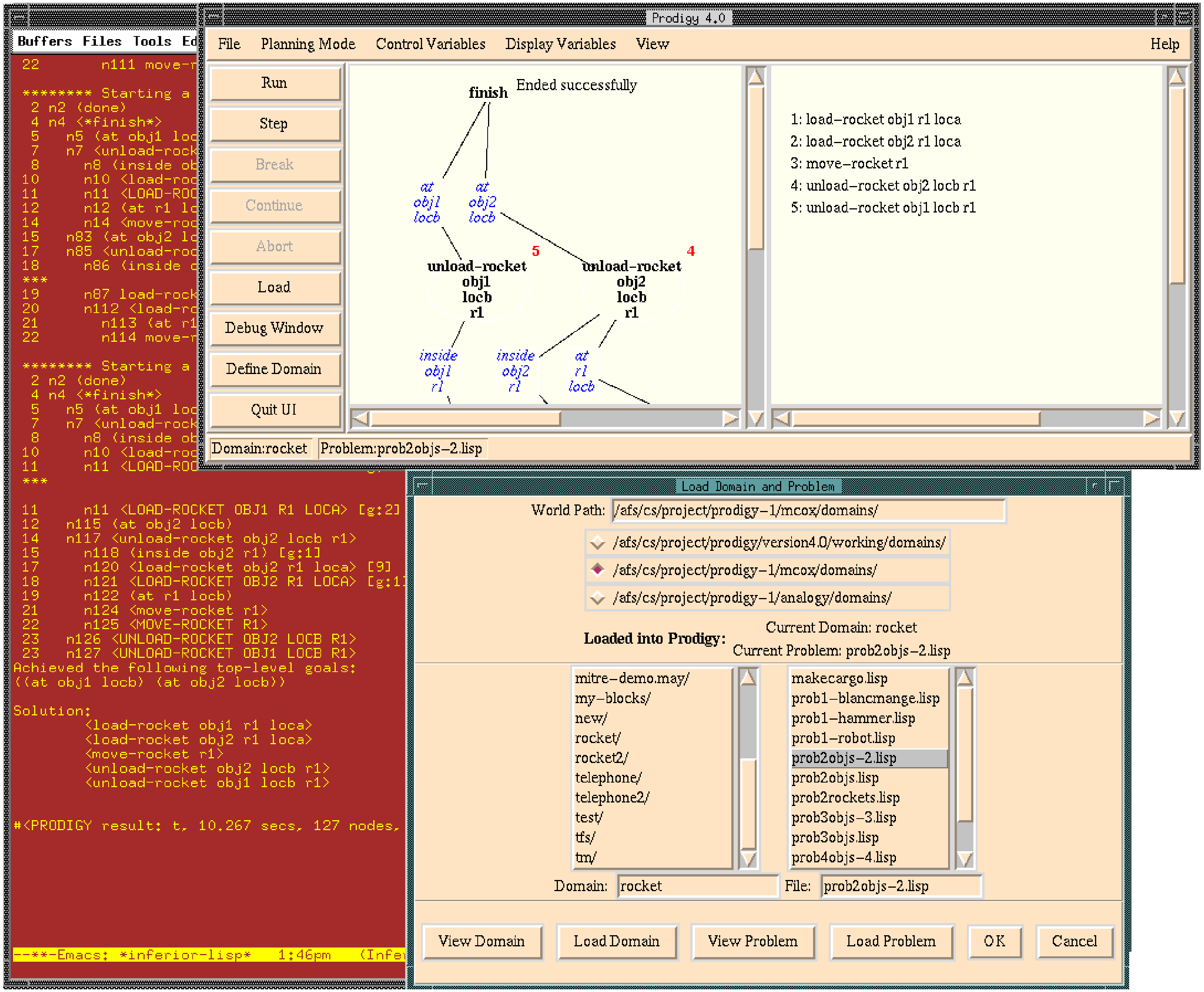
Solution: The **PRODIGY** interface

1. Allows the user to follow both generative and case-based planning.
2. Supports visualization of both plan and the planning rationale.
3. Addresses the variance in the experience of the user by allowing the user to control the presentation of information.

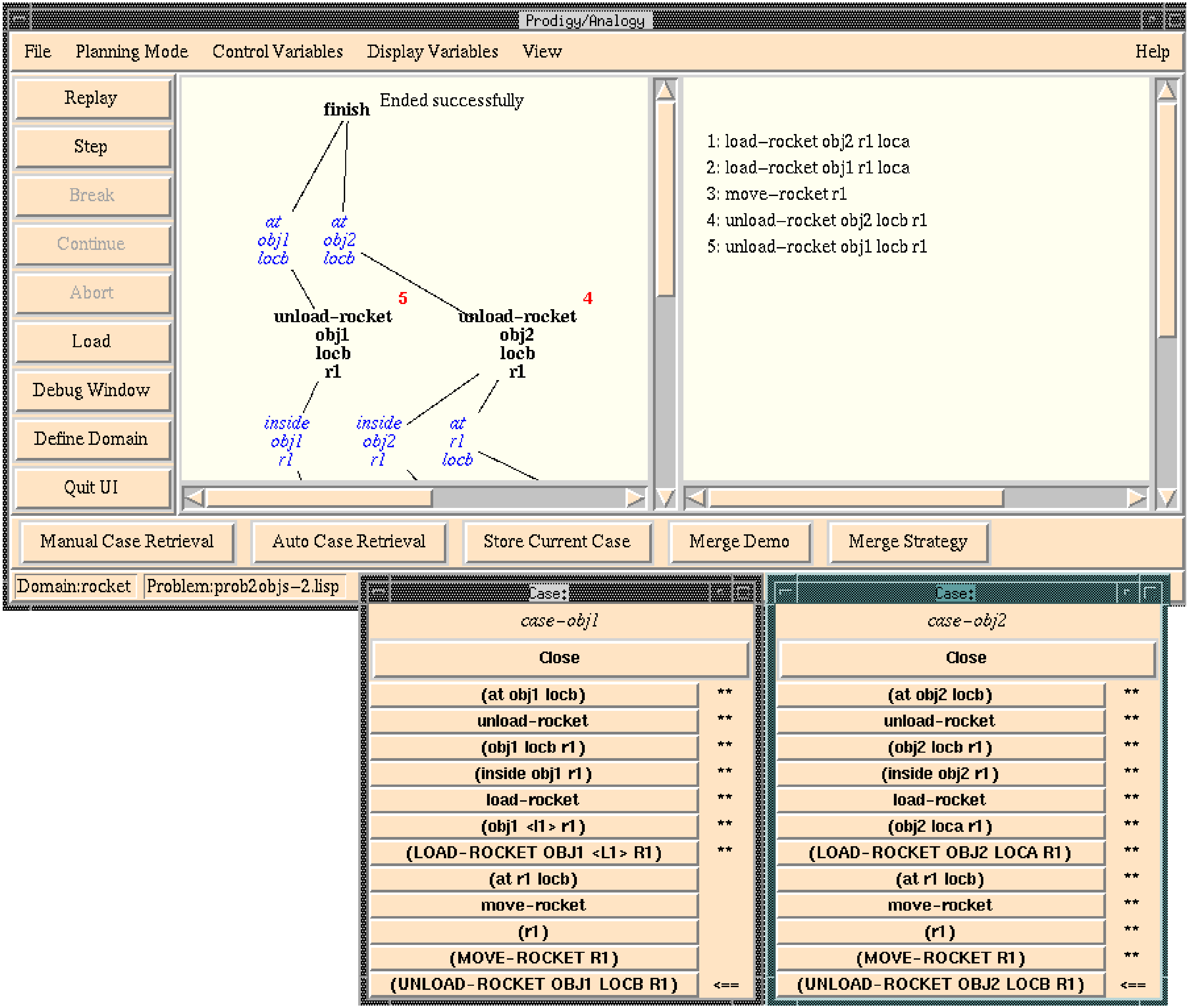
Old Interface (Lisp)

```
; Loading /afs/cs/project/prodigy-1/analogy/domains/rocket/domain.lisp.
Subgoal behaviour: SUBGOAL-ALWAYS-BEFORE-APPLY
;   Loading /afs/cs/project/prodigy-1/analogy/replay-crs.lisp.
Reading Meta predicate: analogy-get-guidance-goal
Reading Meta predicate: analogy-get-guidance-operator
Reading Meta predicate: analogy-get-guidance-bindings
Reading Meta predicate: analogy-decide-subgoal
Reading Meta predicate: analogy-decide-apply
0.0
USER(67): (problem ‘prob2objs-2)
; Loading /afs/cs/project/prodigy-1/analogy/domains/rocket/probs/prob2objs-2.lisp.
T
USER(68): (use-cases 2 0.6 “case-prob1-blancmange” “case-prob1-hammer” “case-prob1-robot”)
NIL
USER(69): (init-guiding)
List of available cases :
 0: “CASE-PROB1-ROBOT.15036”
 1: “CASE-PROB1-BLANCMANGE.15038”
 Current *merge-mode* is SABA
NIL
USER(70): (run)
Creating objects (LOCA LOCB) of type LOCATION
Creating objects (OBJ1 OBJ2) of type OBJECT
Creating object R1 of type ROCKET

  2 n2 (done)
  4 n4 <*finish*>
  5   n5 (at obj2 locb)
  7   n7 <unload-rocket obj2 locb r1>
  8     n8 (inside obj2 r1)
 10     n10 <load-rocket obj2 r1 loca>
 11   n11 (at obj1 locb)
 13   n13 <unload-rocket obj1 locb r1>
 14     n14 (inside obj1 r1)
 16     n16 <load-rocket obj1 r1 loca>
 17     n17 <LOAD-ROCKET OBJ1 R1 LOCA> (“CASE-PROB1-BLANCMANGE.15038” 10)
 18     n18 (at r1 locb)
 20     n20 <move-rocket r1>
 21     n21 <LOAD-ROCKET OBJ2 R1 LOCA> (“CASE-PROB1-ROBOT.15036” 10)
 22     n22 <MOVE-ROCKET R1> (“CASE-PROB1-BLANCMANGE.15038” 14)
 23   n23 <UNLOAD-ROCKET OBJ1 LOCB R1> (“CASE-PROB1-BLANCMANGE.15038” 7)
 23   n24 <UNLOAD-ROCKET OBJ2 LOCB R1> (“CASE-PROB1-ROBOT.15036” 7)
Achieved the following top-level goals:
((AT OBJ1 LOCB) (AT OBJ2 LOCB))
Solution:
    <load-rocket obj1 r1 loca> (“CASE-PROB1-BLANCMANGE.15038” 10)
    <load-rocket obj2 r1 loca> (“CASE-PROB1-ROBOT.15036” 10)
    <move-rocket r1> (“CASE-PROB1-BLANCMANGE.15038” 14)
    <unload-rocket obj1 locb r1> (“CASE-PROB1-BLANCMANGE.15038” 7)
    <unload-rocket obj2 locb r1> (“CASE-PROB1-ROBOT.15036” 7)

#<PRODIGY result: T, 0.7 secs, 24 nodes, 1 sol>
```

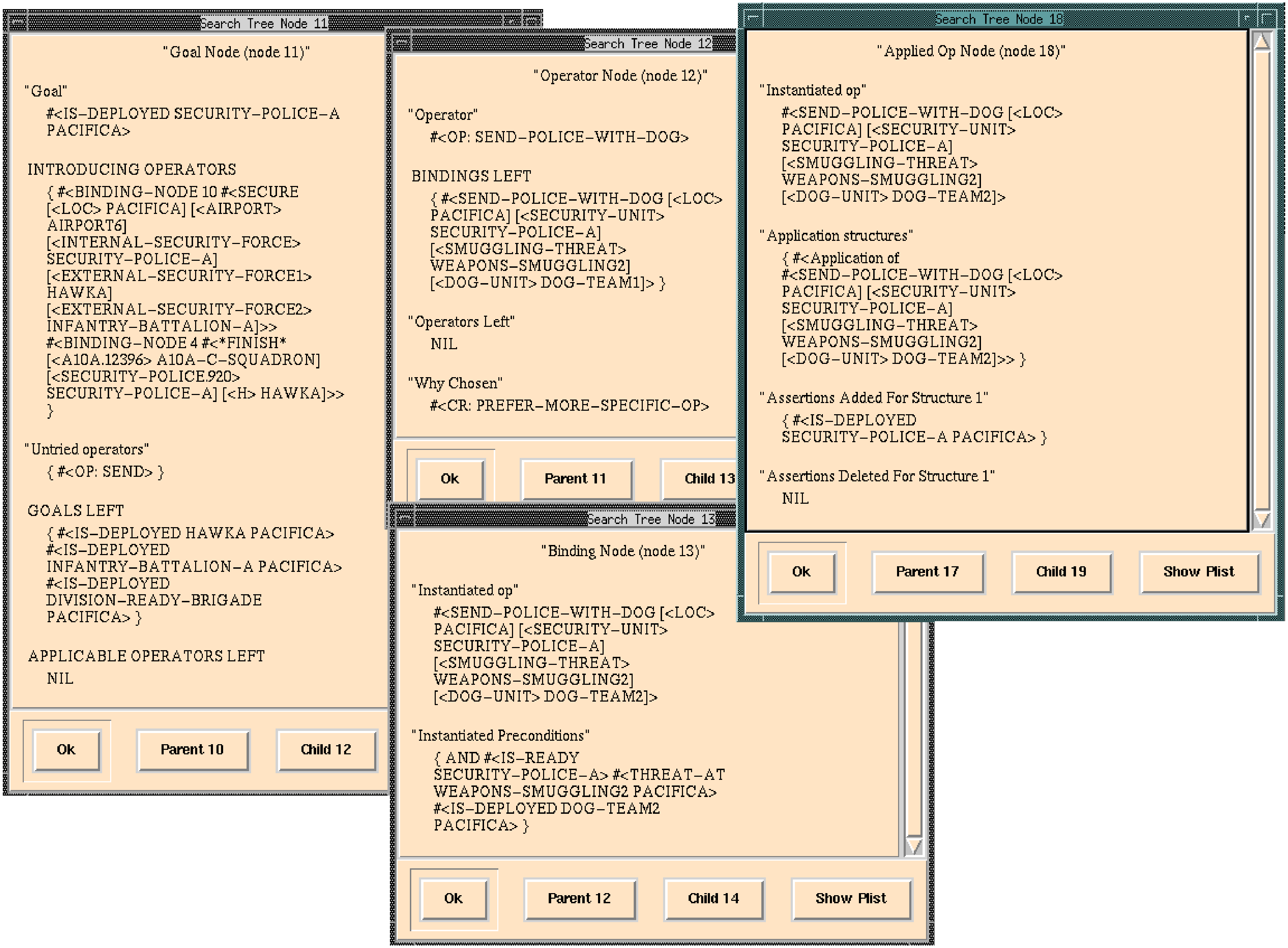
New GUI Interface



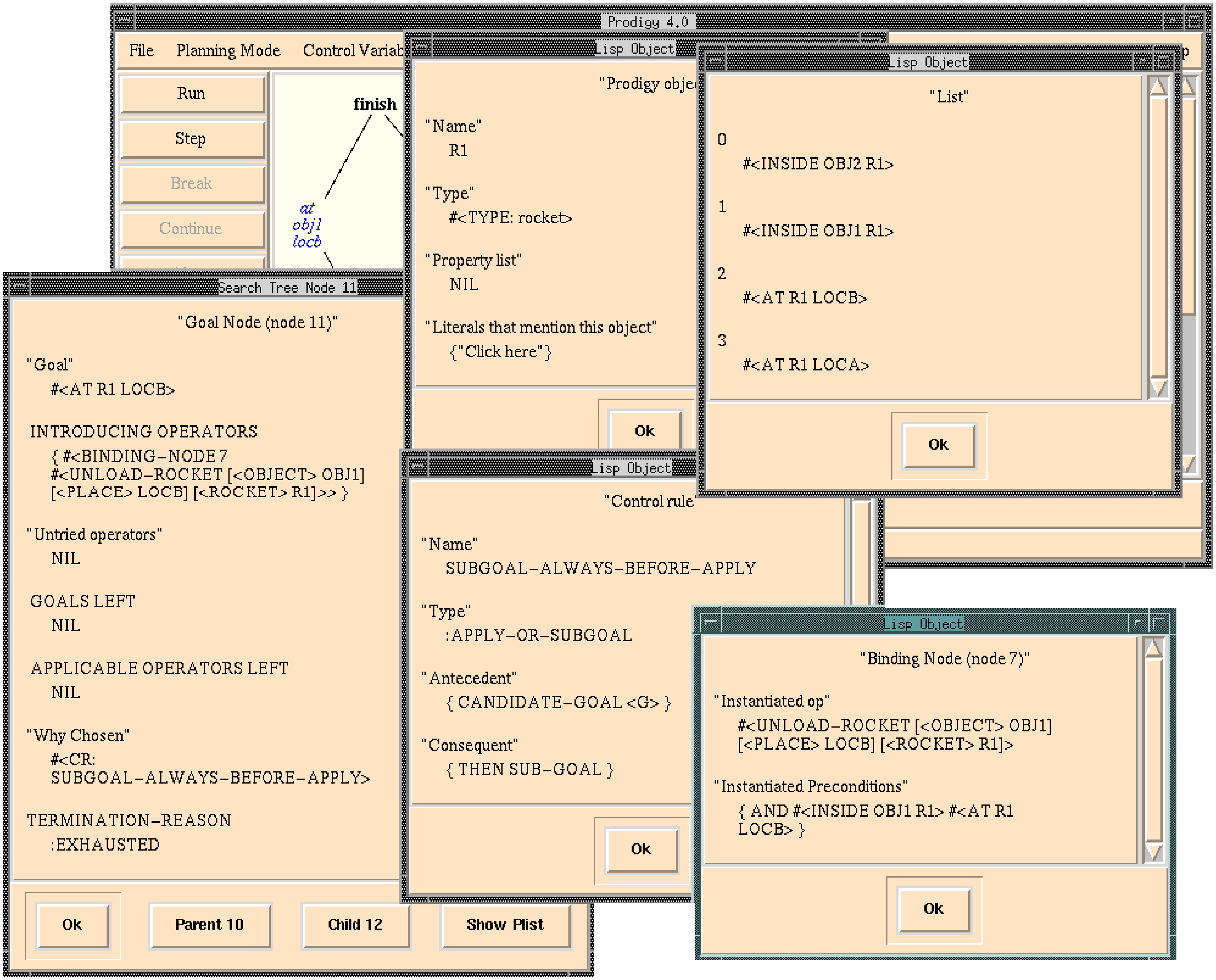
Replaying Past Cases from User Interface



All Planning Decisions can be Examined



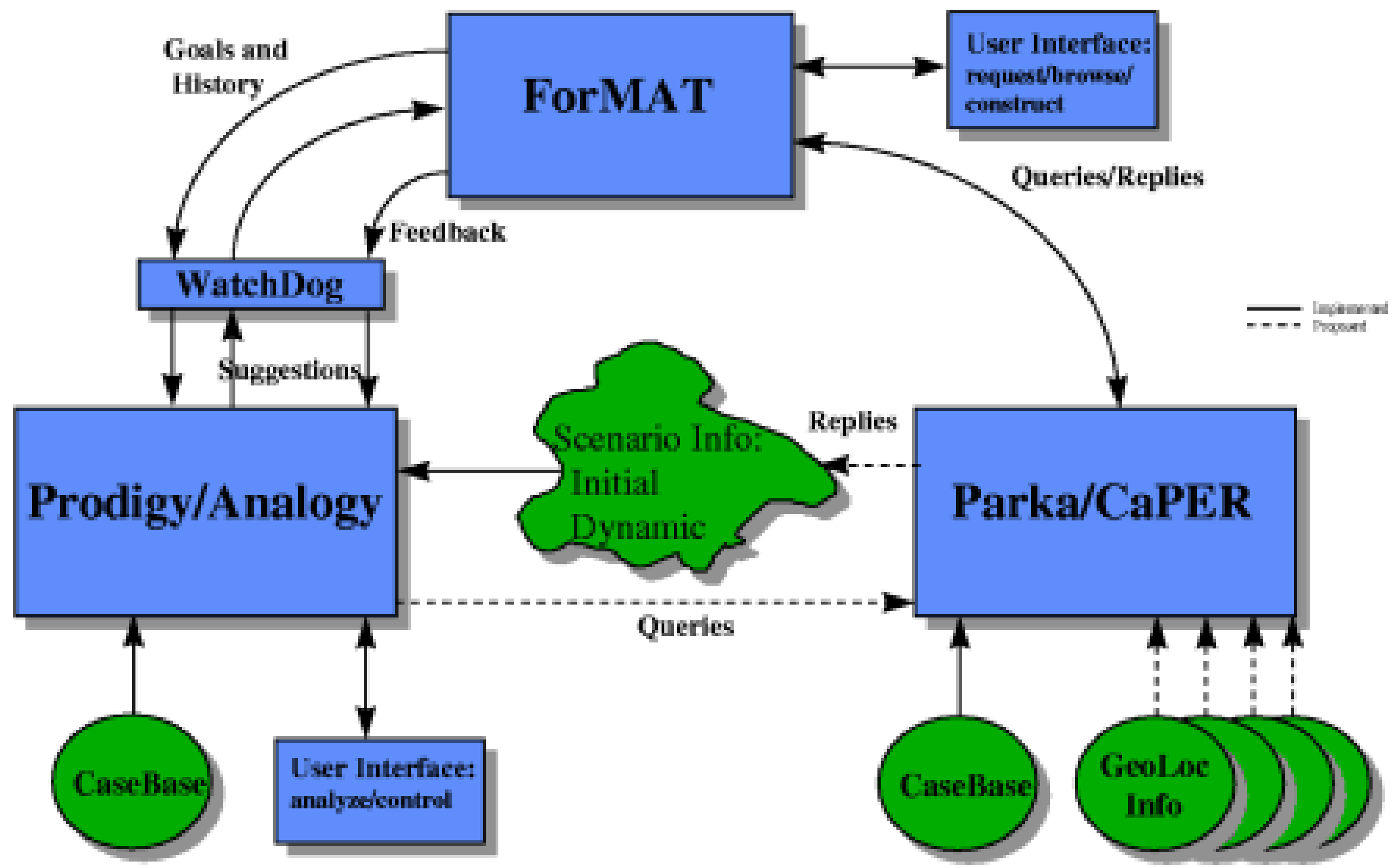
Drill-Down Inspections



Mixed-Initiative Case-Based Planning: MI-CBP

Problems with Goal Expressibility

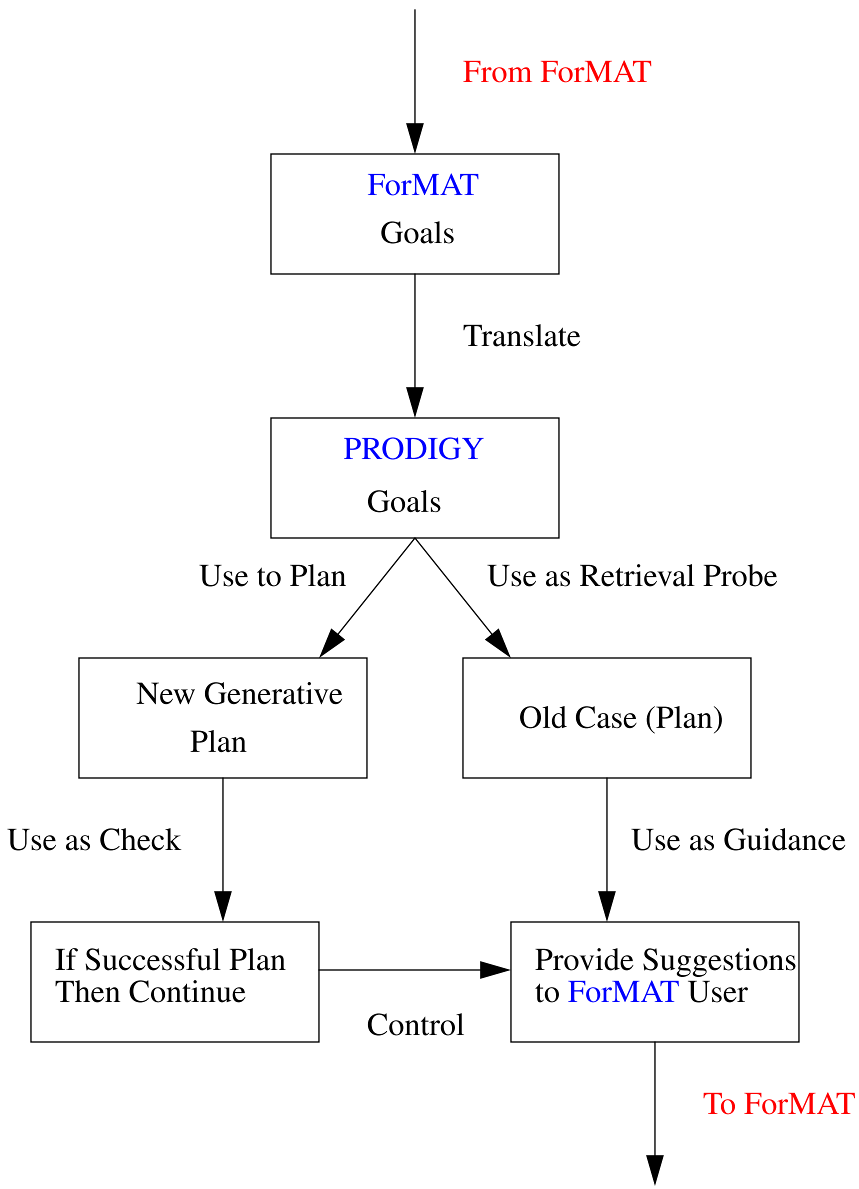
MI-CBP Architecture



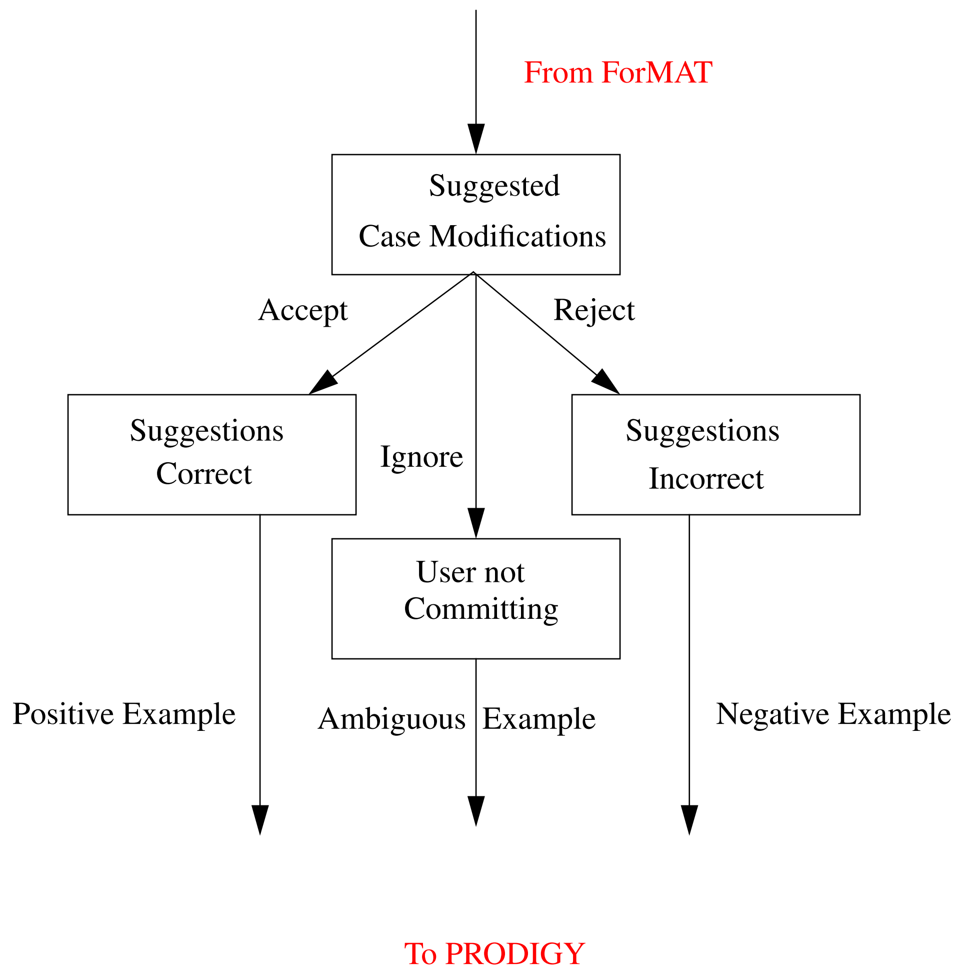
ForMAT supports the following capabilities:

- User receives the text description of a new mission;
- User queries database of past plans;
- User browses past plans;
- User refines query incrementally;
- User uses objective representation to specify the goals of the new mission;
- Plans and their Force Modules (FMs) are stored, *indexed by goals* and scenario information.

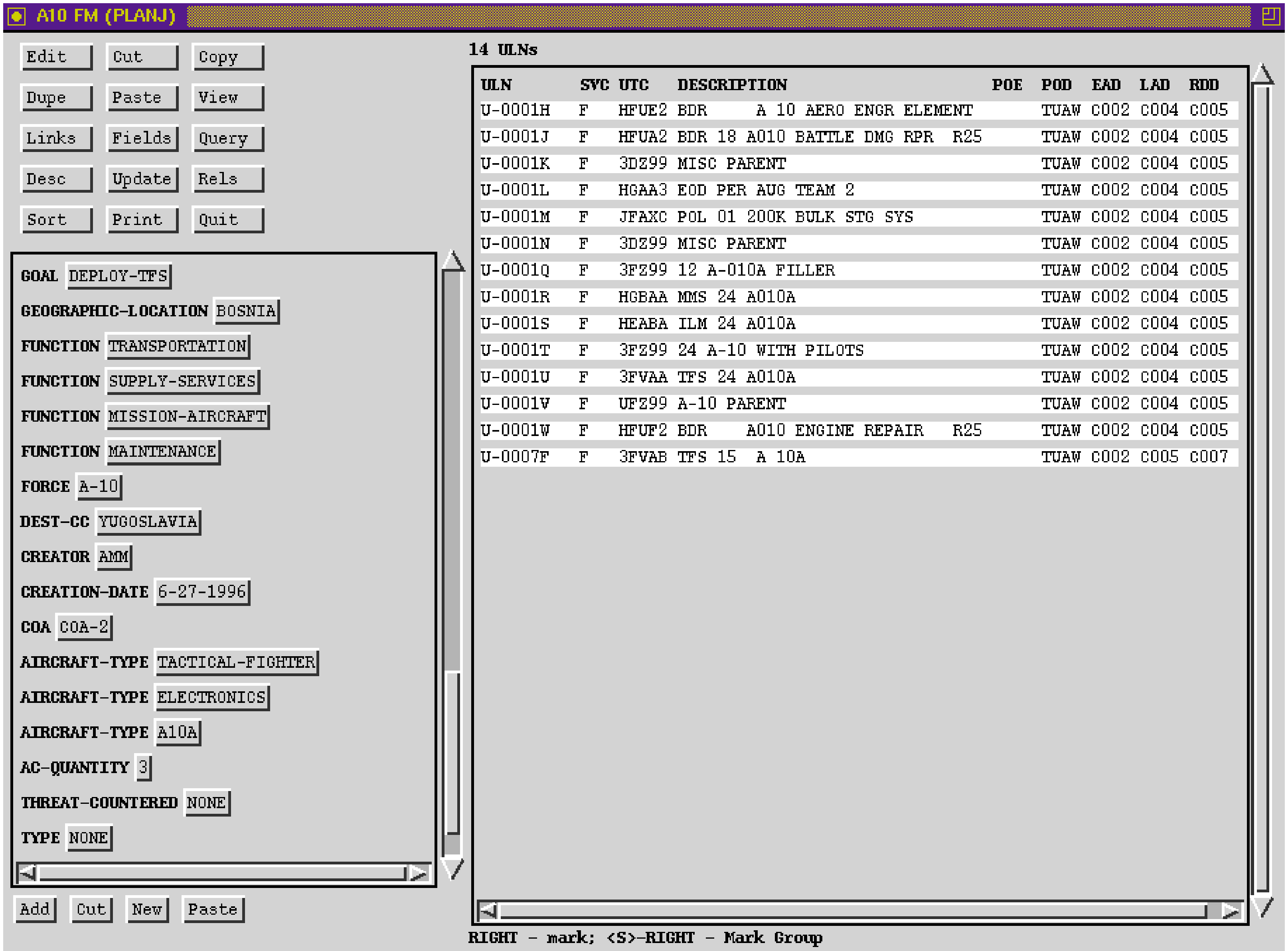
Feedforward Control



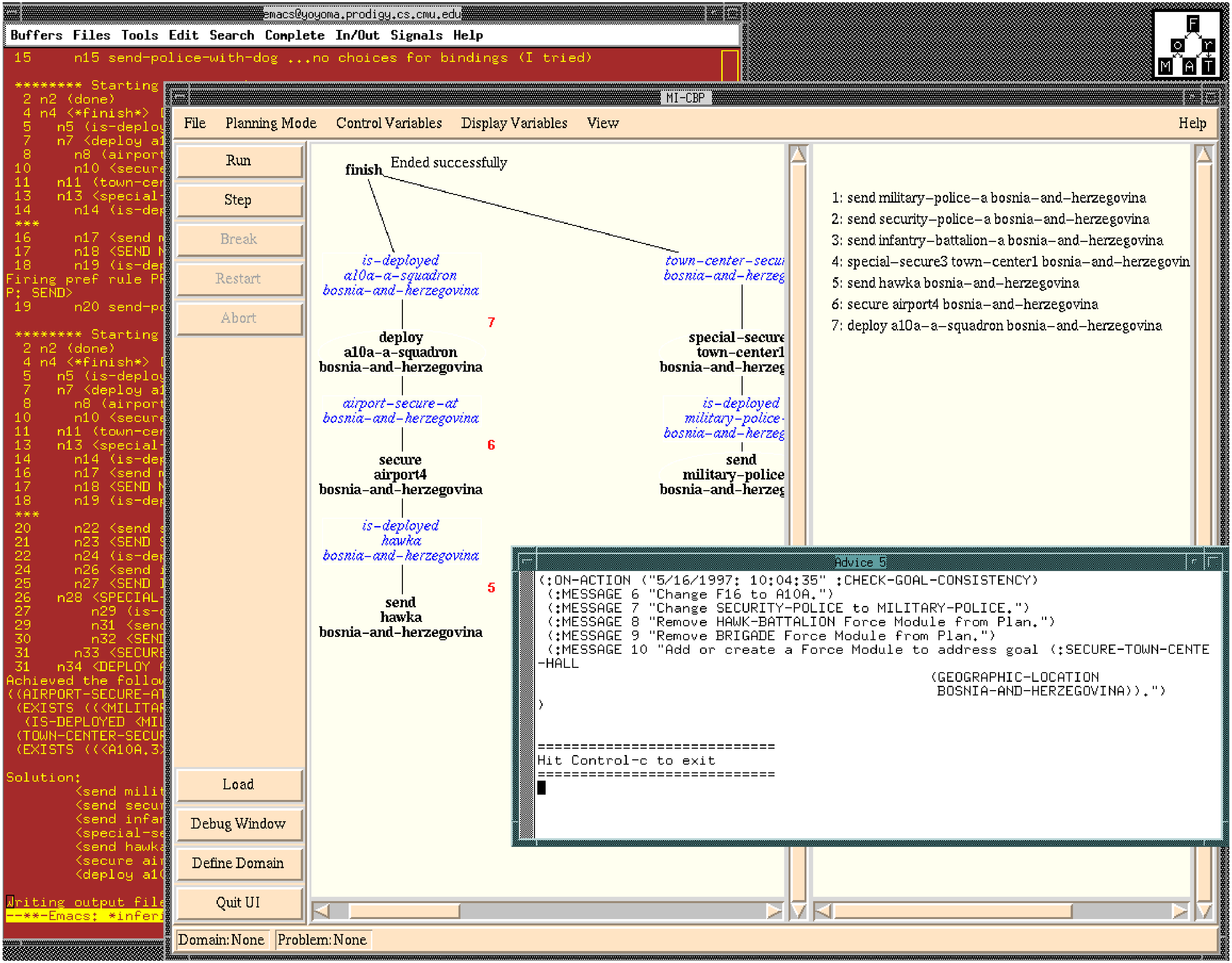
Feedback Control



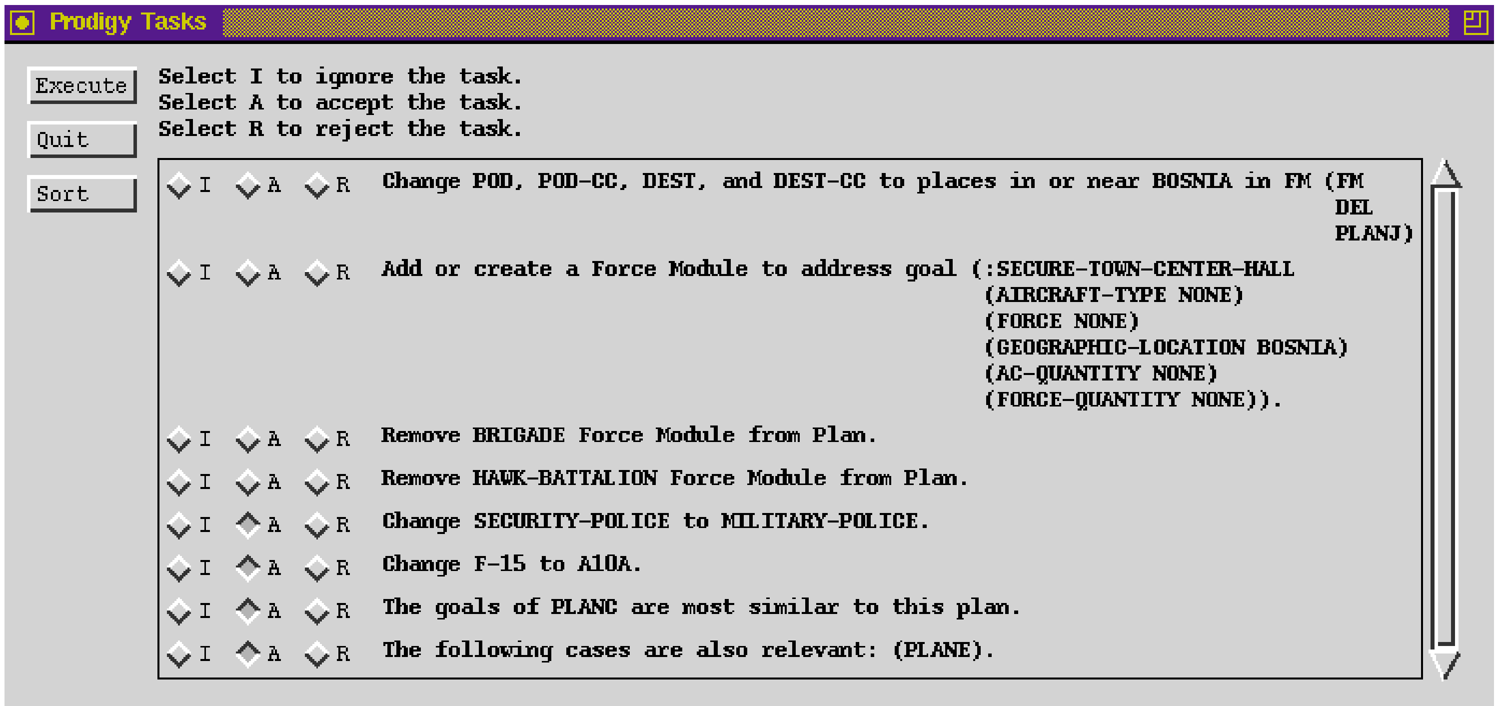
The ForMAT Goal Editor



Prodigy Sends Case Modification Suggestions



The Suggestions from the ForMAT User's Perspective



Problems with unrestricted goal input

1. Users may want to achieve goals in terms of actions as well as states.
2. Users may specify goals that vary along a dimension of abstraction and specificity.
3. Users may mix both top-level goals and subgoals when describing what they want a plan to do.

Control Heuristics:

1. When input desired actions instead of state, preprocess by transformation.

2. When input includes abstract goals, choose operator that is most specific.

3. When input includes both top-level and subordinate goals, achieve top-level goals first but opportunistically select variable bindings that achieve subordinate goals as well.

Commander's Mission Statement

Need a Hawk unit and the 21st Division Ready Brigade to send to Pacifica to secure an airport. Also want to provide security police to keep the airbase secure so that a squadron of A-10As can be forward deployed there.

Example ForMAT Input

```
(:GOALS
(g-146 :SEND-SECURITY-POLICE
        (GEOGRAPHIC-LOCATION PACIFICA))   )
(g-145 :SEND-BRIGADE
        ((FORCE 21ST-BRIGADE)
         (GEOGRAPHIC-LOCATION PACIFICA))   )
(g-144 :SEND-HAWK
        ((FORCE HAWK-BATTALION)
         (GEOGRAPHIC-LOCATION PACIFICA))   )
(g-143 :DEPLOY-A10A
        ((GEOGRAPHIC-LOCATION PACIFICA)
         (AIRCRAFT-TYPE A-10A)) )   )
```

Goals as Actions

Problem 1: User may provide goal as action to be carried out rather than state (objective) to achieve

Solution:

1. Parse goal and arguments
2. Determine whether state or action
3. If action then

    Lookup corresponding operator
    Post associated primary effect.
    Remember desired action

```
(g-145 :SEND-BRIGADE
        ((FORCE 21ST-BRIGADE)
        (GEOGRAPHIC-LOCATION PACIFICA))   )
--> (is-deployed 21ST-BRIGADE PACIFICA)
```

Goal and Operator Hierarchies

Problem 2: User may not specify grounded literals.

Solution: Existentially quantified goals

```
(g-144 :SEND-HAWK
            ((FORCE HAWK-BATTALION)
            (GEOGRAPHIC-LOCATION PACIFICA))   )
--> (exists ((<hwk> hawk-battalion))
        (is-deployed <hwk> PACIFICA))
```

But, given two operators that can achieve the goal along different points in the abstraction hierarchy, which to choose?

Solution: Choose the more specific one.

Operator **secure** is More General Than **secure-airport**

secure-airport (<obj>, <loc>)
<loc>: type LOCATION
<obj>: type AIRPORT
<internal-security>: type
POLICE-FORCE-MODULE
<external-security>: type TROOPS
<air-defense>: type ANTI-AIR-MODULE
Pre: (loc-at <obj> <loc>)
(is-deployed <internal-security> <obj>)
(is-deployed <air-defense> <obj>)
(is-deployed <external-security> <obj>)
Add:(secure-at <loc> <obj>)

Can achieve (secure-at Pacifica Airport2)

secure (<obj>, <loc>)
<loc>: type LOCATION
<obj>: type OBJECT
<internal-security>: type
POLICE-FORCE-MODULE
<external-security>: type TROOPS
Pre: (loc-at <obj> <loc>)
(is-deployed <internal-security> <obj>)
(is-deployed <external-security> <obj>)
Add:(secure-at <loc> <obj>)

*Can achieve both (secure-at Pacifica Airport2)
and (secure-at Pacifica Hill5)*

Example:

(is-deployed Hawk-1 Pacifica) is a descendant of (is-deployed HAWK-BATTALION COUNTRY)

Goal Descendancy

Abbreviation 1: Given G = (predicate arg1 .. argn) *such that* predicate $\in$ ExistingPredicates *and given type hierarchy graph* H = (V, E) *where* V *is the set of class vertices and* E *is the set of incident edges* <x y> *such that* x *is a member of the class set* y. *Then* G *above is an abbreviation for the expression*

$$\forall\, arg_{i=1..n} \mid arg_i \in V, \exists\, x_i \mid x_i \in arg_i, (\text{predicate } x_1\ x_2\ ..\ x_n).$$

Definition 1: *Given that* Arg(G, i) *is the ith argument of goal* G *as abbreviated above*, *we define goal* G_1 *to be an **ancestor** of* G_2 (G_2 *is a **descendant** of* G_1) *in hierarchy* H *iff*

(1) $arity(G_1) = arity(G_2) = n$
(2) $predicate(G_1) = predicate(G_2)$
(3) for i=1..n, $Arg(G_1, i) \geq Arg(G_2, i)$ *in* H

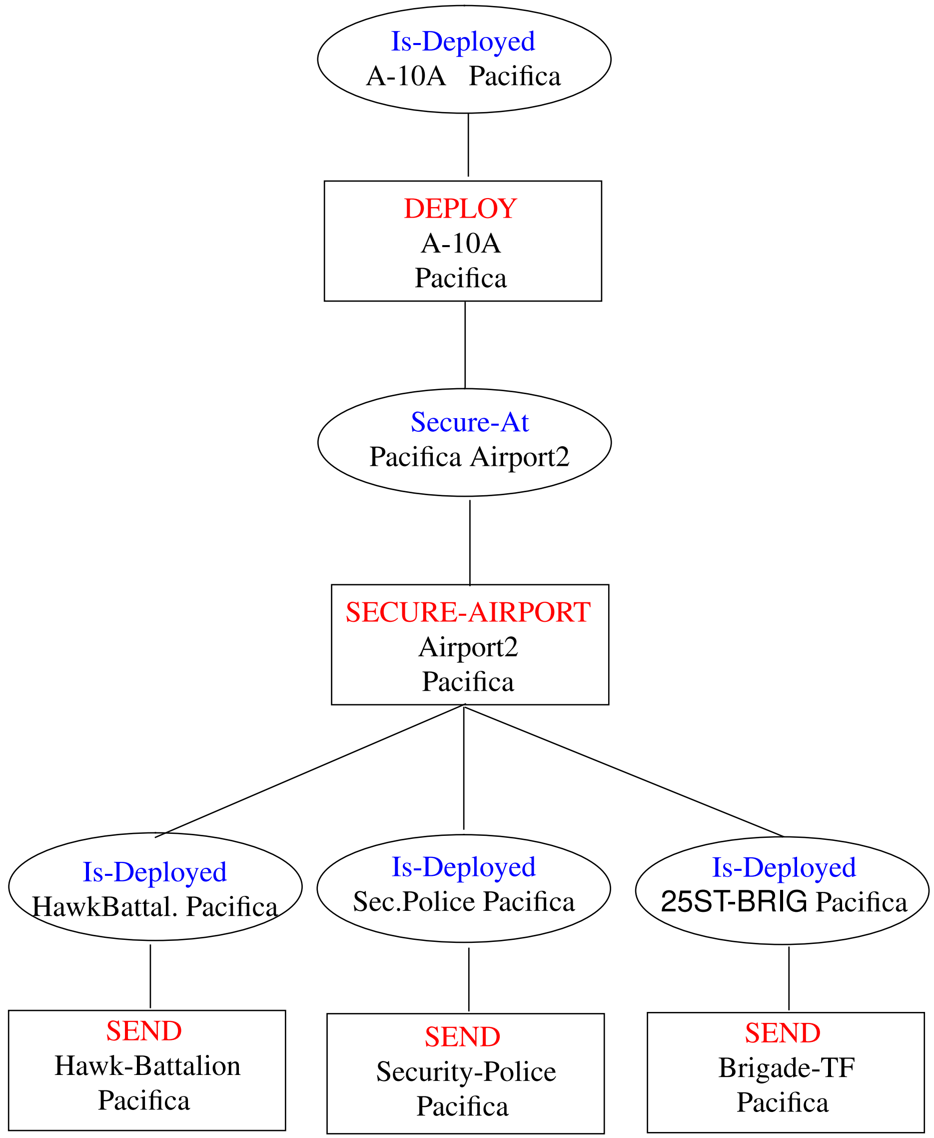
where the relation "≥" between arguments is defined as equivalence or class inheritance such that a path exists through H *from* $Arg(G_2, i)$ *to* $Arg(G_1, i)$. *That is, the edges* $<Arg(G_2, i)\ e_1>\ <e_1\ e_2>\ ...\ <e_k\ Arg(G_1, i)>$ *are all elements of edge list* E.

PRODIGY Control Rule

```
(CONTROL-RULE Prefer-More-Specific-Op
  (if (and (candidate-operator <OP1>)
           (candidate-operator <OP2>)
           (is-ancestor-op-of-p <OP1> <OP2>)))
  (then prefer operator <OP1> <OP2>) )
```

Given two applicable operators from which to choose, prefer the more specific one

PRODIGY Goal Tree



Top-Level Goals Versus Subgoals

Problem 3: Given both top-level and subordinate goals by the user, which to solve first.

Solution: Solve top-level first to provide guidance and to reduce back-tracking.

```
(CONTROL-RULE Prefer-Top-Most-Goal
 (if (and (candidate-goal <G1>)
          (candidate-goal <G2>)
          (subgoal-of-p <G2> <G1>)))
  (then prefer goal <G1> <G2>) )
```

Given two goals, prefer one if making the other true solves one of the preconditions for an operator that results in the preferred one

But planning top-down, operator bindings may be inconsistent.

Solution: Opportunistic planning

Example:

(is-deployed HAWK-BATTALION Pacifica) is a subgoal of (is-deployed A-10-A Pacifica)

Goal Subordination

Definition 2: G_2, *is a* ***subgoal*** *of* G_1, *(or alternatively,* G_1 *is a* ***supergoal*** *of* G_2*)*
iff

(1)*a backward operators chain exists* $OP_1 .. OP_n$ *from* G_1 *to* G_2 *such that* $\forall OP_{i=1..n-1}\ \exists p,e \mid p \in pre(OP_i), e \in eff(OP_{i+1})$, p *unifies with* e *under substitution* σ

(2)$\exists p \mid p \in pre(OP_n)$, *and* p *unifies with* G_2 *under* σ

(3)$\exists e \mid e \in eff(OP_1)$, *and* e *unifies with* G_1 *under* σ .

In the case of $n=1$, a single operator, OP, exists whose effects includes the effect $e \in eff(OP)$ that unifies with G_1 under the simple substitution, σ, and whose preconditions includes $p \in pre(OP)$ that unifies with G_2 also under σ.

PRODIGY Control Rule

```
(CONTROL-RULE Prefer-Bindings-Opportunistically
(if (and(current-operator <OP>)
        (candidate-bindings <CB>)
        (match-constraining-goals <G> <OP>)
        (generate-new-bindings <NB> <G> <OP>)
        (identify-worse-bindings <CB> <NB> <WB><OP>)
        (identify-better-bindings <CB> <NB> <BB><OP>)))
(then prefer bindings <BB> <WB>))
```

Given current operator and candidate set of bindings, prefer bindings that opportunistically solve another pending goal

Opportunities

- Unexpected goals represent user-provided hints on how to construct good plans.
- Abstract goals allow the system to avoid overfitting the plan.
- Subgoal constraint information that users provide can originate outside the transitive closure of a domain theory.

Research Contributions

- Formal specification of both goal ascendancy and goal subordination and related operator hierarchy.
- Control heuristics for managing three nonstandard user goal specifications:

  1. When input desired actions instead of state, preprocess by transformation.

  2. When input includes abstract goals, choose operator that is most specific.

3. When input includes both top-level and subordinate goals, achieve top-level goals first but opportunistically select variable bindings that achieve subordinate goals as well.

- The challenge involved with mixed-initiative systems is to engineer a more robust mechanism that is flexible with respect to user needs.
- The opportunity is to leverage user experience.

Continuous Mixed-Initiative Planning and Execution:

Replanning as Goal Transformations

Goal Transformation as Movement in a Goal Space

- Over Predicate Abstraction Hierarchy
- Over Argument Abstraction Hierarchy
- Over Argument Component Hierarchy (Partonomy)
- Over Argument Values (Enumerated or Numerical)

A Hierarchy of Goals
(over Argument Abstraction)

(Is-Deployed 21ST-Division-Ready-Brigade Korea)

(Exists ((<B> BRIGADE))
    (Is-Deployed <B> Korea))

(Exists ((<FM> FORCE-MODULE)
    (<L> LOCATION))
    (Is-Deployed <FM> <L>))

A Taxonomy of Goal Transformations

Transformation	Example
Operationalization	acquired (air-superiority) → destroyed (enemy-air-forces)
Concretion	destroyed (air-forces) → destroyed (offensive-air-forces)
Specialization	ineffective (enemy-forces) → destroyed (enemy-forces)
Instantiation	deployed (airborne-unit) → deployed (82nd-Airborne-Div)
Reification	deployed (82nd-Airborne-Div) → Flown (82nd-Airborne-Div)
Expansion	interdicted (rail-line) → interdicted (rail-net)
Contraction	secured (city) → secured (city-airport)
Erosion	fighting-capacity (enemy, 50%) → fighting-capacity (enemy, 75%)
Escalation	outcome (battle, stalemate) → outcome (battle, victory)
Intrusion	null → in-control-of(base)
Retraction	deployed (reconnaissance-unit) → null
Identity	G → G
Substitution	prevent (not(in-control-of(base))) → maintain (in-control-of(base))

The Bridge Problem

- Each river has 3 bridges
- Each plane can destroy 1 bridge
- Each plane can damage any bridge
- To make a river impassable, destroy all bridges accross it
- To make a river restricted, destroy as many as possible and damage the rest
- Destroying a bridge reduces transportation capacity by 100%
- Damaging a bridge reduces transportation capacity by 50%

Goal Operationalization

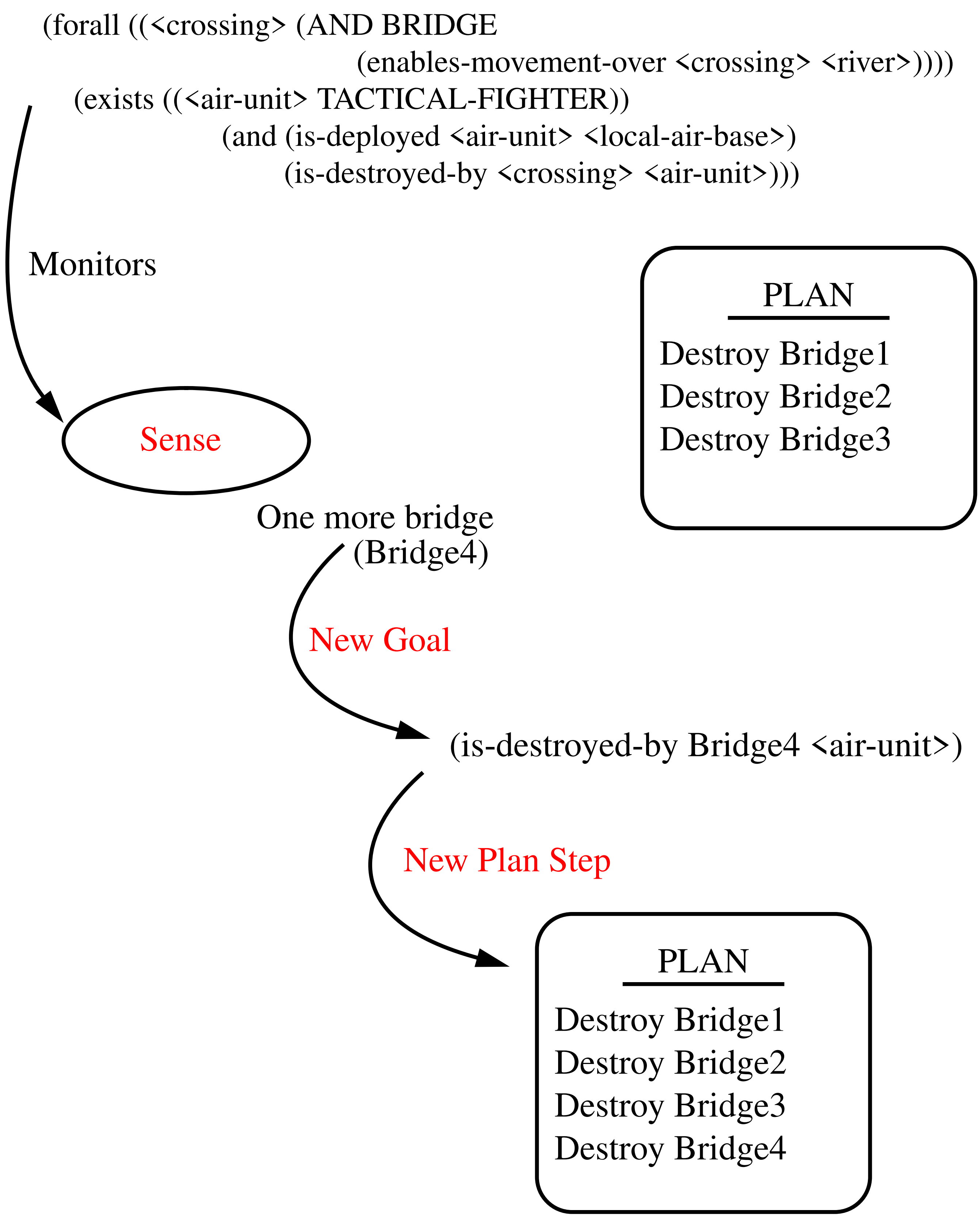
Goal = (outcome impassable Yalu)

$\Downarrow$

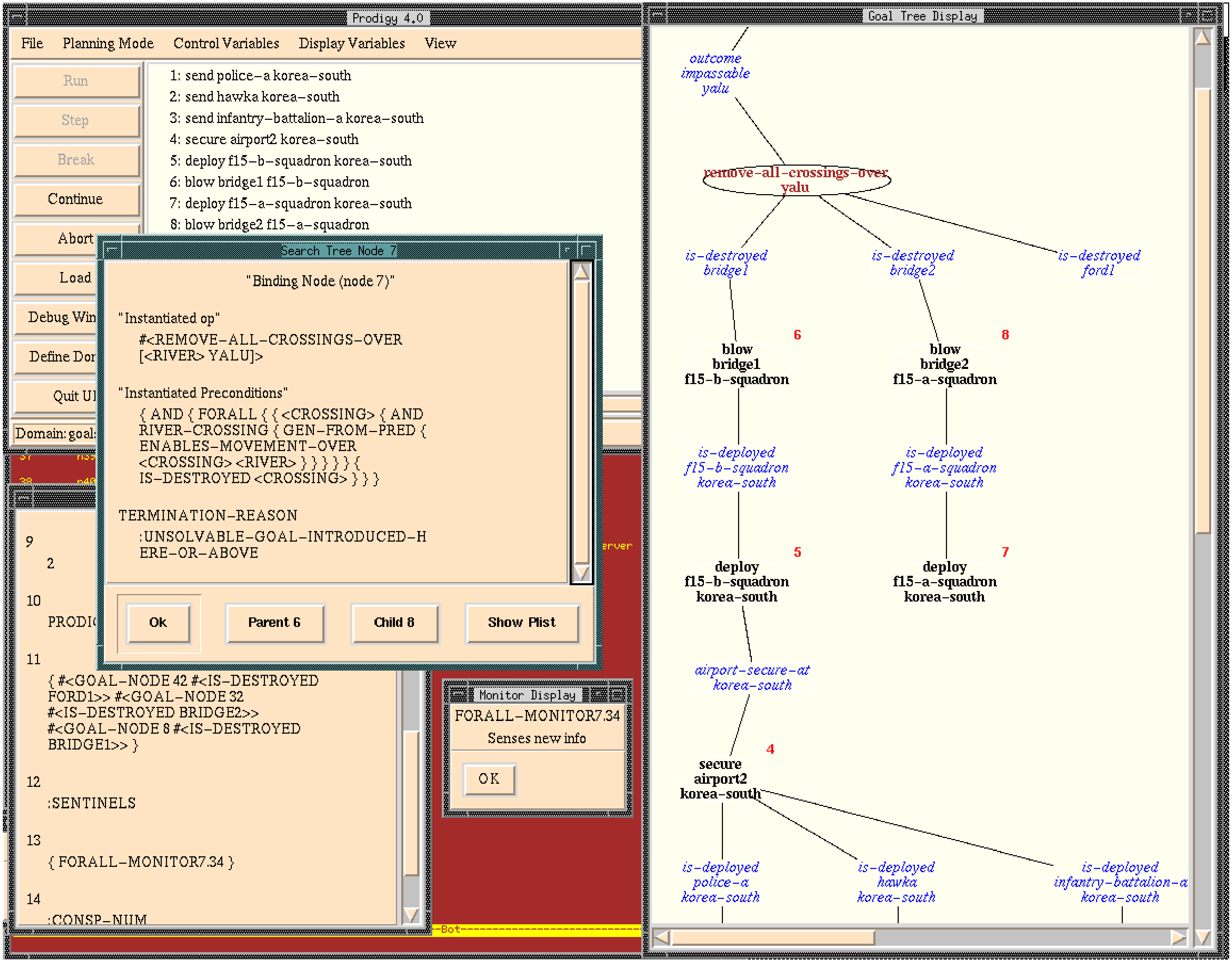
Goals = {(is-destroyed-by Bridge1 F15-A)
(is-destroyed-by Bridge2 F15-B)
(is-destroyed-by Bridge3 F15-C)}

```
(INFERENCE-RULE REMOVE-ALL-CROSSINGS-OVER
 (params <river>)
 (preconds
  ((<river> WATER-BARRIER)
   (<local-air-base> AIRPORT))
  (and
    (near <local-air-base> <river>)
    (is-secure <local-air-base>)
    (forall ((<crossing> (AND BRIDGE
                              (enables-movement-over <crossing> <river>))))
            (exists ((<air-unit> TACTICAL-FIGHTER))
                    (and (is-deployed <air-unit> <local-air-base>)
                         (is-destroyed-by <crossing> <air-unit>))))))
 (effects
  ()
  ((add  (outcome impassable <river>))))
 )
```

Goal Intrusion

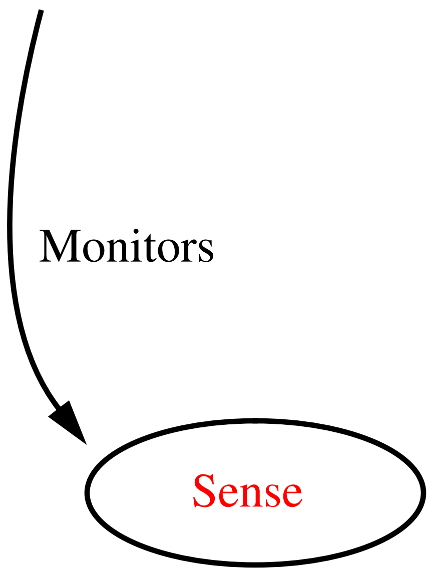


After Goal Intrusion Transformation



Goal Erosion

```
(forall ((<crossing> (AND BRIDGE
                          (enables-movement-over <crossing> <river>))))
   (exists ((<air-unit> TACTICAL-FIGHTER))
           (and (is-deployed <air-unit> <local-air-base>)
                (is-destroyed-by <crossing> <air-unit>)))
```



PLAN

Destroy Bridge1
Destroy Bridge2
Destroy Bridge3
Damage Bridge4
Damage Bridge5

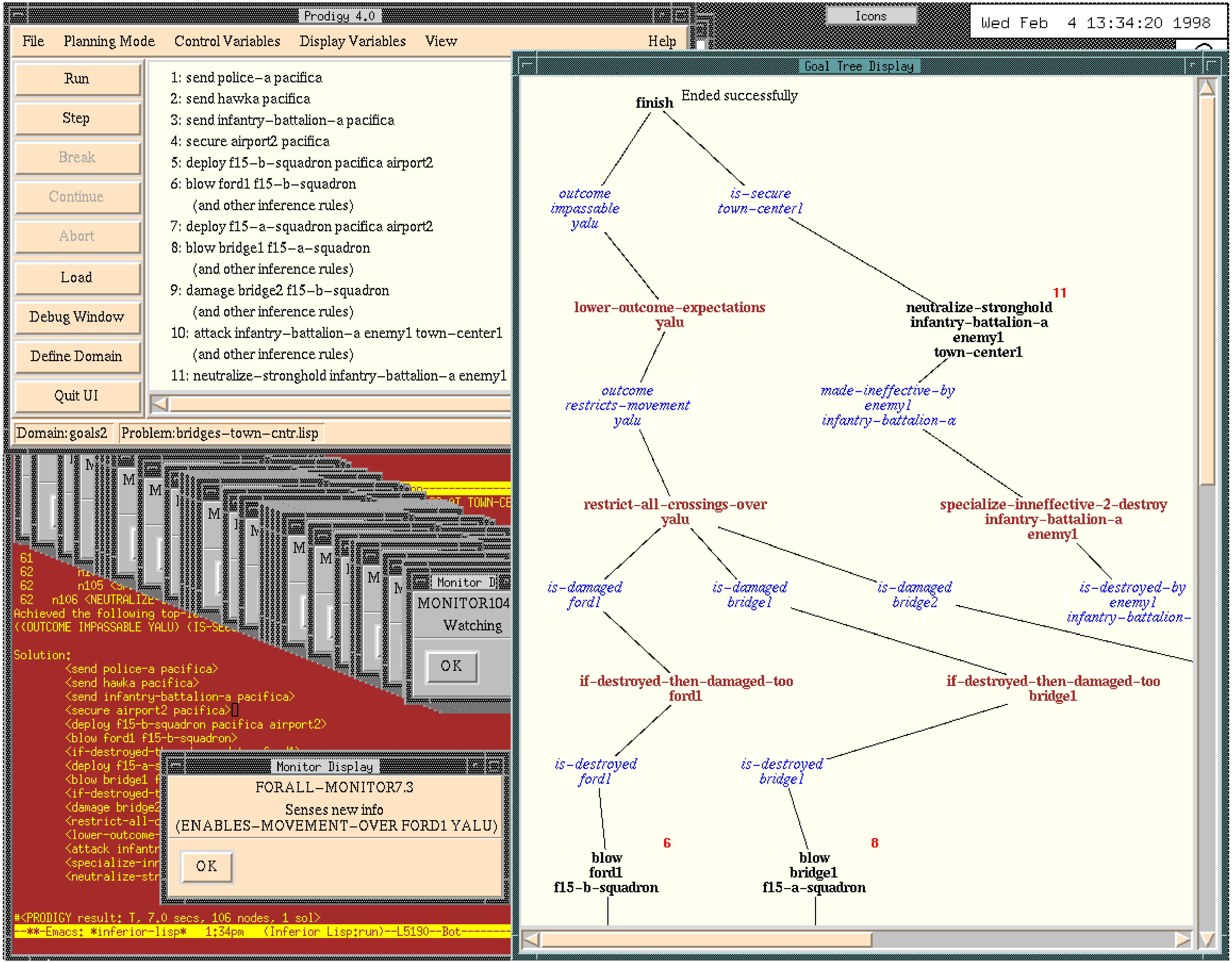
One more bridge
(Bridge5)
\+ Insufficient resources (Tactical fighters)

Goal = (outcome impassable Yalu)

$\Downarrow$

Goal = (outcome restricts-movement Yalu)

After Goal Erosion Transformation



Independent Variables:

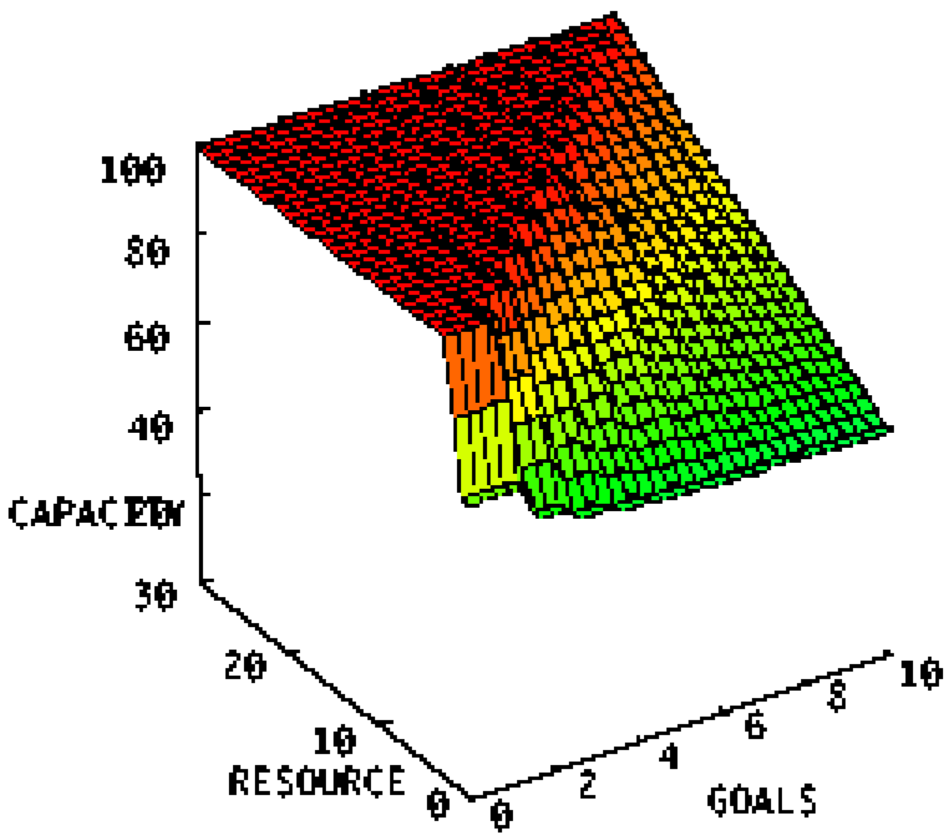
- Resources (Planes)
- Goals (Rivers)

Dependent Variable

- Reduction in Transportation Capacity

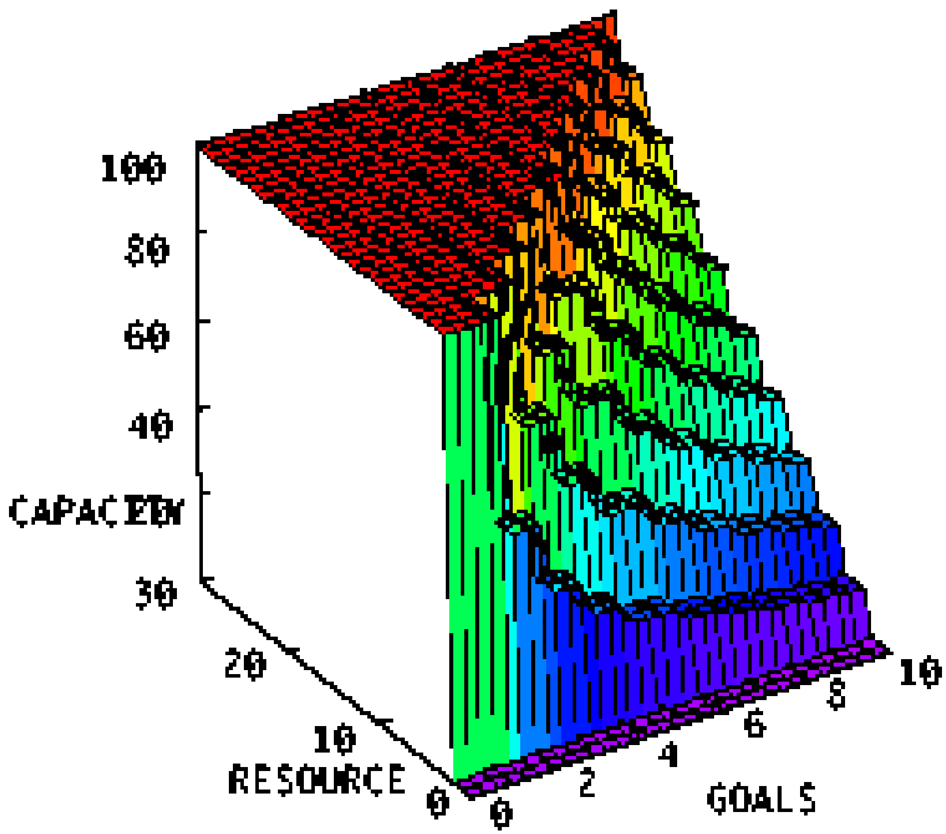
Plan performance with goal transformations

(as a function of resource availability and problem complexity)

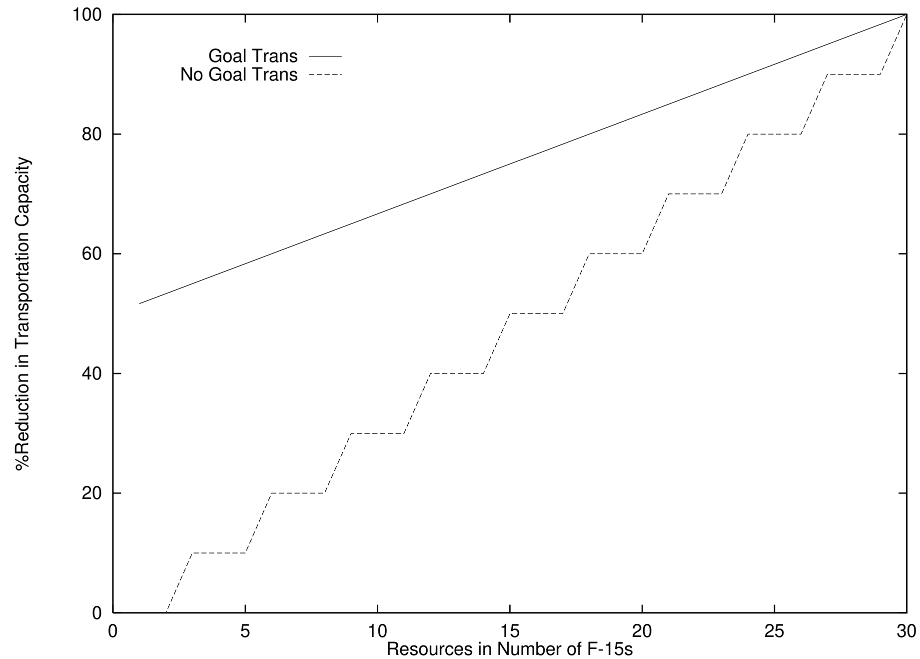


Performance without goal transformations

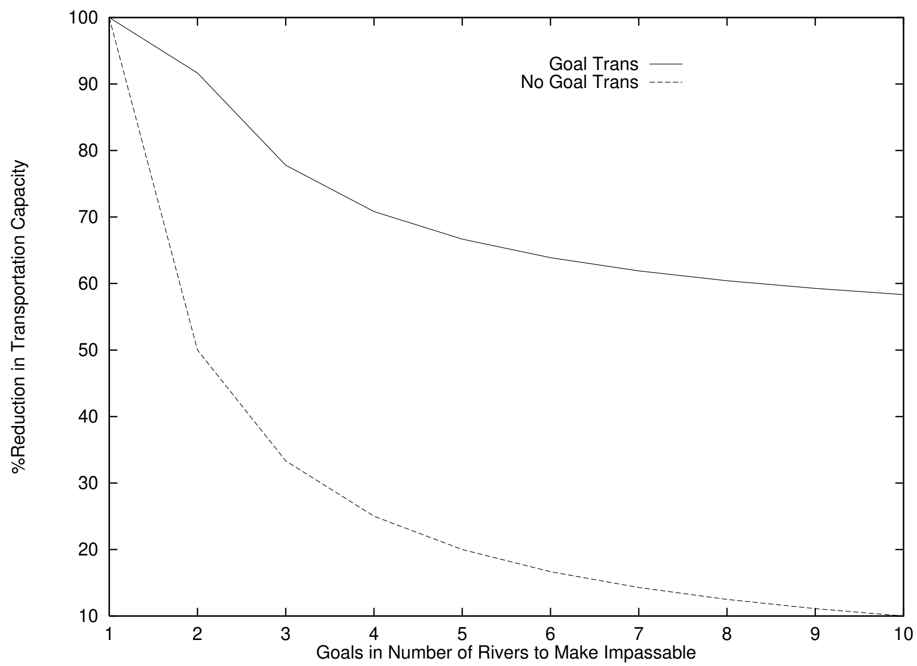
(as a function of resource availability and problem complexity)



Plan performance as a function of resource availability given 10 goals.



Plan performance as a function of problem complexity given 5 resources.



Research Contributions

- Planning generates *sensing monitors* based on the plan rationale.
- Replanning in response to a changing environment involves not just plan adaptation, but also the *transformation of objectives*.
- *Control knowledge* is used to guide the plan and goal adaptation choices.

Future Research

- Novice Mode for GUI
- Learning to Improve Modification Suggestions in MI-CBP
- Cost/Benefit Analysis for Goal Transformations
- Interdisciplinary Research with Cognitive Scientists

Conclusion

- Planning is Facilitated by Visualization and User Control
- Goal Representation Needs Redefinition
- Replanning Entails Goal Shifts
- Continuous Planning and Execution Require Rationale-Based Sensing
- Humans and Machines Can Make a Successful Marriage