MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

Working Paper No. 330 May, 1990

An Experiment in Knowledge Acquisition for Software Requirements

Paul M. Lefelhocz

The Requirements Apprentice (RA) is a demonstration system that assists a human analyst in the requirements-acquisition phase of the software-development process. By applying the RA to another example it has been possible to show some of the range of applicability of the RA. The same disambiguation, formalization, and contradiction-resolution techniques are useful in the air traffic control and library database domains and some clichés are shared between them. In addition, the need for an extension to the RA is seen: summarization of contradictions could be improved.

Contents

1 The Requirements Apprentice

A prototype *Requirements Apprentice* (RA) had been developed by Reubenstein. The purpose of the RA is to assist a requirements analyst in developing a requirement for a software system. The RA supports the earliest phases of creating a requirement in which ambiguity, contradiction, and incompleteness are inevitable features. When helping the analyst create a requirement, the RA stores the information it obtains in a *Requirements Knowledge Base* (RKB). The RKB represents all of the information that the RA knows about a requirement. This information can be transformed into a traditional requirements document. The rest of this section gives a short introduction to the RA. However, for more background and details about the RA see [3, 4, 5, 6].

A central component within the RA is its cliché library. A cliché is a commonly occurring structure. Rich argues that clichés exist in most engineering disciplines [2]. The clichés used by the RA are formally codified objects that represent the actual clichés of some discipline. The use of clichés allows domain specific knowledge to be given to the RA as data rather than being built into it. This is what gives the RA its wide-range applicability. The RA can be tailored to new domains by adding clichés relevant to that domain.

The clichés are organized in a hierarchy so that as additional information is obtained about an object, the cliché that applies to it can be specialized. The specialization of the clichés relevant to an object can have the effect of increasing the amount of knowledge in the RKB. For example, if it is known that an object (P) is an instance of the position cliché and that it has a particular longitude, the RA can infer that P is an instance of the cartesian position cliché. Once the RA knows that P is an instance of the cartesian position cliché, it can add the information that P has a latitude and an altitude to the RKB. This has the effect of adding knowledge to the RKB as a direct consequence of specialization of the cliché relevant to P. The process of determining which clichés an object is an instance of is called disambiguation.

Disambiguation is an important process in the RA. When an analyst uses a word that has many different meanings, it is the job of the RA to determine which of them the analyst had in mind. The different meanings of a word are codified by clichés that specialize the cliché representing the general word. Once a specialized meaning is determined, the RA can obtain additional information from the knowledge in the cliché library.

The RA can also detect contradictions. Because CAKE (the reasoner driving the RA [1]) is not a complete reasoner, not all contradictions are determined. However, the RA detects all contradictions that follow from sufficiently simple input and it provides assistance in dealing with these in the form of showing the deductions used to determine a contradiction, suggesting premises to be substituted for premises underlying the contradiction, and allowing the analyst to defer the contradiction

until more information is obtained. Premises are formal codifications of assumptions made by the analyst or the RA.

The RA also detects incompleteness in a requirement and allows support for an evolving requirement. Detecting incompleteness is useful to remind the analyst of forgotten details, and supporting an evolving requirement allows the analyst a way to correct mistakes.

Reubenstein's demonstration of the RA deals with an evolving requirement for a university library database system and a legal example. The most important clichés in his example are the environment, needs, system, tracking system, and information system clichés. The environment, system, and needs clichés allow the RA to support a theory of requirements that says requirements are composed of three main kinds of information: environment, system, and needs. The environment describes the context of the requirement - those aspects not under the control of the end user, the needs describe the desires of the end-user, and the system describes a high-level specification of a system that meets the needs.

A tracking system keeps track of the state of some object or set of objects by making observations. It can make these observation directly (e.g., the way a radar keeps track of a set of aircraft) or indirectly (e.g., the way a set of turnstiles keeps track of the number of people in a building). Also, tracking systems can track either single targets or multiple targets. If a tracking system tracks multiple targets, the targets must already be matched with their respective data (i.e., there is not only a set of data and a set of targets but a map between the two). If the targets and data are unmatched, a multiple target tracking system is unable to track the target set as desired and some other part of the system will have to determine the map between target and data. An information system describes a class of programs that create, modify, delete, and report data.

The particular domain discussed in this paper is air traffic control (See Section 2). The relevant clichés to the domain are the environment, system, needs, tracking system, display system, trajectory and position clichés. Some of these are the same as in the library database domain. A display system is used to display data. The display might be a screen, printer, or another output device. A trajectory is a succession of data over time. This can be the historical record of the states of some object or the predicted states of an object. A position describes the location of an object. It can be specified in many different coordinate systems (e.g., polar, cartesian, or cylindrical).

The ATC example shows that the RA can be applied to other domains, there is the potential for sharing of clichés between domains, and the intersection of high level clichés can produce useful lower level clichés (See Section 3). Also, the incompleteness of the RA has been addressed: one useful extension would be better contradiction summarization (See Section 4).

2 A Session with The Requirements Apprentice

The domain of this example is air traffic control (ATC). An air traffic control system is a system that assists air traffic controllers in tracking and scheduling aircraft. The system takes input from a radar and displays that information to the controller. It also notifies the controller of certain events. For example, the controller is notified if two aircraft are close enough to each other that they might collide or if an aircraft strays too far from its proposed path (flight plan).

The tracking system cliché is relevant to the air traffic control domain because the radar tracks a set of aircraft; those in the observation region specified for the system. The region of observation for a tracking system is specified by a volume. Thus, the volume cliché is also relevant to air traffic control. Like a position, a volume can be specified in many different coordinate systems.

A monitoring system compares two sets of data (e.g., a system could monitor the average grades and the grades of a particular student on a series of exams) and reports information about the comparison. (e.g., The system could report if a student fell more than 10 points below the average.) Since controllers need to be notified if the distance between an aircraft and it's flight plan is too large, the ATC system needs to monitor aircraft and their flight plans. This means that the monitoring system cliché is relevant to air traffic control.

The need to describe flight plans shows that trajectories are relevant to air traffic control. A flight plan is an ordered set of proposed positions of an aircraft. The ordering is given by time: for each time, the desired position of the aircraft is given. Since a trajectory is an ordered set of data over time, we see that a flight plan is a trajectory. Also, the historical record of where an aircraft has been is a trajectory.

The air traffic control system must also display its data to a controller. Hence the need for a display system when considering air traffic control. A display system not only displays a set of data, but also insures that the desired tolerances on the displayed data are obtainable.

A cliché has slots called roles that are filled with objects for particular instances of the cliché. The values of the roles of a cliché instance define the instance of the cliché. Clichés also contain constraints that apply restrictions to the roles. For example, the type of a particular role might be constrained (e.g., the latitude of an instance of the position cliché might be constrained to be a number with associated units), or some relationship between two roles might be specified (e.g., for a cubic volume, the latitude, longitude, and altitude are constrained to be equal).

Using Clichés

To begin the requirement for an air traffic control system, the requirements analyst gives it a name, Aircraft-System, and tells the RA a requirement is being started.

This is seen in command 1. The input to the RA is in the form of calls to a Lisp macro. This is not the optimal form for the user interface of the RA, but the research on the RA is centered on issues other than user interface, hence a command language was deemed sufficient. A proposal for a more sophisticated user interface is given in [5]. The scenario described below is excerpted from an actual session with the RA. The RA prompts the analyst with a number specifying which command is about to be entered.

```
1 --> (Ra> (Find-Requirement Aircraft-System))
Beginning-A-New-Requirement-Called-The Aircraft-System.
Aircraft-System Is-An-Instance-Of Requirement.
```

In response to the analysts commands, the RA announces changes in the RKB. The are two kinds of information returned: paraphrasatic and deductive. The paraphrasatic output is an echo of the information given directly by the command. It allows the analyst to ensure that the RA has correctly interpreted the statement. Most of this output will not be shown throughout the scenario. The second kind of information, deductive, shows the analyst the information the RA has inferred from the commands given. The response for statement 1 tells the analyst that the Aircraft-System is a new requirement.

Statement 2 defines aircraft to be a kind of physical object that has roles the trajectory and flight plan. While aircraft has roles, it should be noted that aircraft is not a cliché. The information given by the analyst here is too impoverished. Were the aircraft a cliché, the definition in the cliché library would give much more detail about aircraft and particular types of aircraft (e.g., aircraft have passengers, need pilots in order to fly, commercial aircraft have a maximum altitude, etc.) However, for this example, we have assumed that the cliché library in use has not been tailored to the ATC domain and hence does not have a cliché to represent aircraft. By putting the trajectory and flight plan roles on aircraft, the analyst creates two functions from the type aircraft to the type trajectory.

```
2 --> (Ra> (Define Aircraft
                :Ako Physical-Object
                :Member-Roles ((The-Trajectory Trajectory)
                               (Flight-Plan Trajectory))))
...
```

The analyst intends for the object in the trajectory role of aircraft to be a historical record of the positions the aircraft was in, and the object in the flight plan role to give its proposed path. However, this information has yet to be specified.

In command 3, the analyst defines a radar that is a tracking system. It tracks a set of Aircraft.

```
3 --> (Ra> (Define Radar :Tracking-System
              :Roles (:Target (!Set-Of* Aircraft))))
Radar Is-A-Member-Of Aircraft-System.The-System.
Radar Is-An-Instance-Of Multiple-Target-Tracking-System.
...
```

As a result of this command, the radar becomes part of the system of the aircraft system (i.e., the radar is part of what will help satisfy the needs of the aircraft system). Also, the radar is a multiple target tracking system. It is multiple target because it tracks a set of objects, not just a single object.

In command 4, the analyst specifies that the radar only tracks the aircraft that are located in a certain area. This area is called an observation region and can be specified in many different forms. The analyst here has decided to specify the region by giving the maximum horizontal distance (radius) and altitude in which the radar makes observations. The center of this region is left unspecified. From this incomplete specification of the observation region of the radar, the RA can determine that the region is an instance of the cylindrical volume cliché.

```
4 --> (Ra> (Fill-Roles Radar.Observation-Region :Max-Radius (Miles 50)
                       :Max-Altitude (Feet 40000)))
Radar.Observation-Region Is-A-Member-Of
  Aircraft-System.The-Environment.
Radar Is-An-Instance-Of
  Multiple-Target-Direct-Observation-Tracking-System.
Radar.Observation-Region Is-An-Instance-Of Cylindrical-Volume.
Radar.Observation-Region.Max-Altitude Has-Value (!Meters 12400.0).
Radar.Observation-Region.Max-Radius Has-Value (!Meters 81840.0).
...
```

From the RA's output, the analyst can see that the RA has converted the measurements given by the analyst to a standard form (in this case miles and feet are converted to meters). Also, it has determined that the radar observes the aircraft directly in some region. This means that the radar has the type multiple target direct observation tracking system.

Detecting Contradictions

In statement 5, the analyst specifies how accurate the data given by the radar is. Since the radar observes polar coordinates, these inaccuracies are ranges on the range, azimuth, and theta of the radar data. From this, the RA infers that the observation region of the radar can be viewed as a polar volume as well as a cylindrical volume. This in turn leads the RA to find the limits of this region in terms of polar coordinates.

Another change to the RKB seen here is that certain objects have the type *limited*. An object being limited specifies that the object, or some part of the object,

has a region it should be in. For example, the radar is limited because the aircraft it observes are only those in the observation region. This is not reported because it is synonymous with the idea of direct observation: any direct observation is made with a sensor that has finite scope.

```
5 --> (Ra> (Fill-Roles
            Radar.Observation.The-Error :Range-Error (Feet 20)
            :Theta-Error (Degrees 1) :Azimuth-Error (Degrees 1)))
Radar.Observation-Region Is-An-Instance-Of Cylindrical-Polar-Volume.
Radar.Observation-Region.Max-Azimuth Has-Value (!Radians 1.57).
Radar.Observation-Region.Min-Azimuth Has-Value (!Radians 0).
Radar.Observation-Region.Max-Range Has-Value (!Meters 116401.59).
Radar.Observation-Region.Min-Range Has-Value (!Meters 0).
Radar.Observation Is-An-Instance-Of Limited-With-Error.
Radar.Observation.The-Error Is-An-Instance-Of Limited-Polar-Error.
Radar.Observation.The-Error.Azimuth-Error Has-Value (!Radians 0.017).
Radar.Observation.The-Error.Theta-Error Has-Value (!Radians 0.017).
Radar.Observation.The-Error.Range-Error Has-Value (!Meters 6.2).
```

In command 6, the analyst defines a display system to display the observed data of the radar.

```
6 --> (Ra> (Define Radar-Display :Display-System
              :Roles (Display-Of Radar.Observation)))
Radar-Display Is-A-Member-Of Aircraft-System.The-System.
Radar-Display Is-An-Instance-Of Display-System.
Radar-Display.Display-Of Has-Value Radar.Observation.
```

In command 7, the analyst specifies how accurate the values displayed need to be. These accuracies are given in terms of the latitude, longitude, and altitude since the display system will display the cartesian position of each aircraft. Since cartesian coordinates are used to specify these accuracies, the RA can infer that viewing the observation region as a cartesian volume is relevant to this system and so it finds the appropriate limits on the latitude, longitude, and altitude. Similarly, the RA now knows that it is relevant to view the observation error as a cartesian error. Appropriate conversions from polar coordinates are performed to determine the error of the observation in terms of cartesian coordinates. From these converted values, the RA determines that the altitude is not know accurately enough to be displayed within the tolerances asked for by the analyst. This results in a contradiction being detected.

```
7 --> (Ra> (Fill-Roles
            Radar-Display.Display.The-Error
            :Latitude-Error (Feet 300) :Altitude-Error (Feet 300)
            :Longitude-Error (Feet 300)))
Radar.Observation-Region Is-An-Instance-Of
  Cylindrical-Cartesian-Volume.
Radar.Observation-Region Is-An-Instance-Of Polar-Cartesian-Volume.
Radar.Observation.The-Error Is-An-Instance-Of
  Limited-Polar-Cartesian-Error.
Radar.Observation.The-Error.Altitude-Error Has-Value
  (!Meters 1978.84).
Radar.Observation.The-Error.Longitude-Error Has-Value
  (!Meters 35.21).
Radar.Observation.The-Error.Latitude-Error Has-Value (!Meters 36.79).
Radar-Display.Display.The-Error Is-An-Instance-Of
  Limited-Polar-Cartesian-Error.
Radar-Display.Display.The-Error.Altitude-Error Has-Value
  (!Meters 93.0).
Radar-Display.Display.The-Error.Longitude-Error Has-Value
  (!Meters 93.0).
Radar-Display.Display.The-Error.Latitude-Error Has-Value
  (!Meters 93.0).
```

The RA has determined both that the altitude error of the observed data of the radar is less than 93 meters, and not less than 93 meters. (i.e., The analyst wishes for the data to be displayed within an accuracy of 93 meters, but the data given by the radar is not that accurate.)

At this point, the analyst remembers that the altitude of an aircraft is reported to the aircraft system by a beacon that transmits data from the altimeter of the aircraft. Because of this, the altitude error value converted from the polar error values and the size of the observation region is no longer needed. The beacon will have its own error value. Thus, the analyst retracts the equality node that causes this conversion. Then, in command 8, the analyst defines a beacon from aircraft to data values and gives its error tolerance.

```
There-Are 1 Pending-Contras
A-Contradiction-From-A Modus Ponens Forward Clause-Between->
(= (< (Number-Of (Altitude-Error Radar.Observation.The-Error))
      (Number-Of (Altitude-Error Radar-Display.Display.The-Error)))
   (< (Number-Of (Altitude-Error Radar.Observation.The-Error)) 93.0))
(< (Number-Of (Altitude-Error Radar.Observation.The-Error))
   (Number-Of (Altitude-Error Radar-Display.Display.The-Error)))
(Not (< (Number-Of (Altitude-Error Radar.Observation.The-Error))
        93.0))
...
The-Interesting-Premises-Are
  (Number-In-Parens-Greater-Than-One-Is-Contras-Premise-Is-In)
1. (User= (Latitude-Error (The-Error (Display Radar-Display)))
          (Meters 93.0))
2. (User= (Observation Radar) (Display-Of Radar-Display))
3. (User= (Range-Error (The-Error (Observation Radar))) (Meters 6.2))
4. (User= (Radians 0.017)
          (Azimuth-Error (The-Error (Observation Radar))))
5. (User= (Theta-Error (The-Error (Observation Radar)))
          (Radians 0.017))
6. (User= (Max-Altitude (Observation-Region Radar)) (Meters 12400.0))
7. (User= (Max-Radius (Observation-Region Radar)) (Meters 81840.0))
8. (Display-System Radar-Display)
9. (Tracking-System Radar)
10. (!Default
     8
     (= (Altitude-Error Radar.Observation.The-Error)
        (Meters
         (Rat->Alt-Acc
          (Number-Of (Range-Error Radar.Observation.The-Error))
          (Number-Of (Azimuth-Error Radar.Observation.The-Error))
          (Number-Of (Theta-Error Radar.Observation.The-Error))
          (Number-Of
            (Max-Range
              (Observation-Region Radar.Observation.The-Error)))
          (Number-Of
            (Min-Azimuth
              (Observation-Region Radar.Observation.The-Error)))
          (Number-Of
            (Max-Theta
              (Observation-Region Radar.Observation.The-Error)))))))
```

```
Would you like to retract a contra premise?(Y or N) Yes.
Retract premise # (0 Is None and Loops to all)? 10
 You-Chose
 (!Default
   8
   (= (Altitude-Error Radar.Observation.The-Error)
      (Meters
        (Rat->Alt-Acc
          ...))))
 Resolved-Contradiction
 Lost-Value (!Meters 1978.84) For
   Radar.Observation.The-Error.Altitude-Error.

 8 --> (Ra> (Define Beacon :Data-Source :Roles
                   (:Error-Value (Feet 100) :From Aircraft)))
 ...
```

In command 9, the analyst equates the beacon error value to the altitude error of the radar. This specifies that the radar will use the information from the beacon rather than the converted information. The beacon information is accurate enough for the display system to make a display within the given tolerances. If it were not, the analyst would have to inform the end user and either the display would have to be less accurate or a more accurate source of obtaining information would have to be found.

```
9 --> (Ra> (Fill-Role Radar.Observation.The-Error.Altitude-Error With
                      Beacon.Error-Value))
Radar.Observation.The-Error.Altitude-Error Has-Value (!Meters 31.0).
```

In command 10, the analyst continues the requirements acquisition process by defining the aircraft monitoring system (AMS). While the radar dealt with the data observed about aircraft, the monitoring system deals with the proposed paths of the aircraft, their flight plans as well as the record of the data observed about the aircraft, their trajectories. The AMS will help controllers keep the aircraft on course. Also, it will help the controller determine such problems as a flight plan being proposed that will bring two aircraft too close together.

```
10 --> (Ra> (Define Aircraft-Monitoring-System :System :Synonym Ams))
Ams Is-A-Member-Of Aircraft-System.The-System.
Ams Is-An-Instance-Of System.
```

In command 11, the analyst specifies that the AMS monitors two sets. The first set is the set of trajectories of the set of aircraft. Mathematically, this is the image of the set of aircraft under the function trajectory. The second set is the set of flight plans of the aircraft.

One object of the monitoring system is called its *actual*, the other its *predicted*. The actual or some aspect of it (in this case the aircraft from which the trajectory is determined) usually has a physical existence whereas the predicted is usually a proposed state for the actual. A monitoring system may monitor many pairs of objects at once or just one. Because the objects monitored by the AMS are sets, the RA determines that the AMS deals with multiple targets.

```
11 --> (Ra> (Need (!Logic (Monitors Ams
                                    (!Image-Of (!Set-Of* Aircraft)
                                               The-Trajectory)
                                    (!Image-Of (!Set-Of* Aircraft)
                                               Flight-Plan)))))
Ams Is-An-Instance-Of Multi-Target-Monitoring-System.
Ams.Compare-Pred Has-Value Monitor=.
Ams.Predicted Has-Value (!Image-Of (!Set-Of* Aircraft) Flight-Plan).
Ams.Actual Has-Value (!Image-Of (!Set-Of* Aircraft) The-Trajectory).
...
```

In command 12, the analyst specifies that the observation region of the AMS is the same as the observation region of the radar. Since the radar deals with the positions of aircraft and the AMS with their trajectories, a contradiction is detected.

The reasoning resulting in this contradiction is as follows: since the observation region of the radar is equal to the observation region of the AMS, they both have the same type. The AMS monitors cylindrical trajectories so they are both cylindrical trajectory volumes and hence their object volumes are cylindrical position volumes. However, the radar tracks aircraft which have position. This means that the volume of the radar is a position volume. Position volumes do not have object volumes so the object volume of the radar's observation region is not a cylindrical position volume. This contradicts the earlier deduction that the object volume of the radar observation region is a cylindrical position volume. This type of contradiction is a type conflict. An object is an instance of two clichés (types) that are incompatible: the radar observation region is a cylindrical trajectory volume and a position volume.

```
12 --> (Ra> (Fill-Role Ams.Observation-Region With
                       Radar.Observation-Region))
...
A-Contradiction-From-A Modus Ponens Clause-Between->
(Implies (And (Cliché-Version Cylindrical-Trajectory-Volume 1)
              (Cylindrical-Trajectory-Volume
                Radar.Observation-Region))
         (Cylindrical-Position-Volume
           (Object-Volume Radar.Observation-Region)))
(And (Cliché-Version Cylindrical-Trajectory-Volume 1)
     (Cylindrical-Trajectory-Volume Radar.Observation-Region))
(Not (Cylindrical-Position-Volume
       (Object-Volume Radar.Observation-Region)))
The-Interesting-Premises-Are
  (Number-In-Parens-Greater-Than-One-Is-Contras-Premise-Is-In)
1. ((14) (User= (Target Radar) (!Set-Of* Aircraft)))
2. ((14) (Tracking-System Radar))
3. ((14) (!Choice User 2 (Limited Radar)))
4. ((14) (!Choice Disambig 6
                  (Direct-Observation-Tracking-System Radar)))
5. ((14) (User= (Observation-Region Radar)
                (Observation-Region Aircraft-Monitoring-System)))
6. ((3) (User= (Max-Radius (Observation-Region Radar))
               (Meters 81840.0)))
7. ((14)
    (Need
     (!Logic
      (Monitors
       Aircraft-Monitoring-System
       (!Image-Of (!Set-Of* Aircraft) Trajectory)
       (!Image-Of (!Set-Of* Aircraft) Flight-Plan)))))
...
Would you like to retract a contra premise?(Y or N) Yes.
Retract premise # (0 Is None and Loops to all)? 5
You-Chose (User= (Observation-Region Radar)
                 (Observation-Region Aircraft-Monitoring-System))
Resolved-Contradiction
...
```

Seeing this conflict, the analyst realizes that since the AMS monitors trajectories and the radar tracks aircraft with position, the volumes corresponding to them are incompatible. However, the volumes are related: the objects of the trajectories of the AMS (positions) must be in the same volume as the positions determined by the radar. This is true because the AMS monitors the aircraft trajectories of the same aircraft which have their positions observed by the radar. Thus, the relationship desired between the observation region of the AMS and radar is for the observation region of the radar (a position volume) to be the same as the object volume of the

observation region of the AMS. Therefore, after retracting the previous statement, the analyst conveys the correct information to the RA with command 13.

```
13 --> (Ra> (Fill-Role Ams.Observation-Region.Object-Volume
                       With Radar.Observation-Region))
Aircraft-Monitoring-System.Observation-Region
  Is-A-Member-Of Aircraft-System.The-Environment.
Ams Is-An-Instance-Of
  Multiple-Target-Direct-Observation-Monitoring-System.
Aircraft-Monitoring-System.Observation-Region
  Is-An-Instance-Of Trajectory-Volume.
...
```

A useful extension to the RA which the previous contradiction brings out is to query the analyst for possible alternative roles for a command which tries to assert an equality between objects with incompatible types. In this case, the AMS.Observation-Region is equated with the Radar.Observation-Region. It is reasonable for the RA to query the analyst as to possible alternative roles that might be filled. These might include roles that are in the same cliché as the one being filled and roles of the role being filled. (In this case, the object-volume role of the AMS.observation-region is the relevant role.) The RA might produce a list of possible alternative roles for the user to chose from.

Detecting Violated Expectations

In the next section of input, the analyst begins to reason about the ordering of certain events and tells the RA about error conditions that the system needs to signal. In command 14, the analyst defines four objects of type event. These objects are anonymous individuals meant to be representative of a class of individuals.

The add trajectory event is the event of an aircraft entering the observation region of the AMS. When this happens, its trajectory should be added to those considered by the AMS. The add flight plan event is the event of the flight plan of an aircraft being added to those being considered by the AMS. Both the aircraft hand in event and the aircraft take off event result in the advisory role for an aircraft being given to our aircraft system. The first because another system has relinquished the role and the second because the aircraft took off from an airport within our observation region.

```
14 --> (Ra> (Define (Add-Traj Add-Fp Aircraft-Hand-In
                       Aircraft-Take-Off)
                     :Event))
...
```

In command 15, the analyst specifies that if an add trajectory occurs then an aircraft hand in or aircraft take off should already have occurred. This codifies the

expectation that both before an aircraft takes off and before it enters from a region monitored by another aircraft system, the aircraft system should be notified. The command also specifies that if an aircraft hand in or aircraft take off occurs, we will have to add the aircraft's trajectory in the near future.

The preceded by strictly relation is used to specify that if its second argument occurs (as specified by the predicate event happens), then its first argument will occur later. It also specifies that if the first argument occurs, the second has already occurred. One of is a multiple arity function that forms an event from a set of events. The event represented by the one of form occurs if and only if one or more of the argument events occurs.

```
15 --> (Ra> (Need (!Logic
                   (Preceded-By-Strictly
                     Add-Traj
                     (One-Of Aircraft-Hand-In
                             Aircraft-Take-Off)))))
...
```

The comes before relation species that the first argument event should occur before the second argument event. Thus, command 16 specifies that before there is an aircraft hand in or aircraft take off, the AMS should already have received the flight plan.

```
16 --> (Ra> (Need
              (!Logic
               (Comes-Before Add-Fp
                             (One-Of Aircraft-Hand-In
                                     Aircraft-Take-Off)))))
...
```

In command 17, the analyst specifies that an error should be signaled when an aircraft appears in the observation region without the AMS having received warning in the form of a flight plan. The means by which this error will be signalled is left unspecified for now.

```
17 --> (Ra> (Need (!Logic
                   (Error-When
                    (And (Event-Happens Add-Traj)
                         (Not (Event-Happens Add-Fp)))))))
...
```

Commands 18, 19, and 20 state that it is an error for two planes to be too close, for two flight plans to be filed that would bring their planes too close, and for an aircraft to be too far from its flight plan. At first, these statements may seem redundant: if two planes are not too far from their respective flight plans and the flight plans are

not too close, then the planes should not be too close. However, there can be planes in the airspace that have not filed flight plans and the problem when a plane strays from its flight plan is far less severe than the one when two planes are close enough that they might collide so different errors will be signalled.

```
18 --> (Ra> (Need (!Logic (Error-When
                           (!There-Exists
                            ((?Aircraft1 Aircraft)
                             (?Aircraft2 Aircraft))
                            Such-That
                            (< (Distance ?Aircraft1.Trajectory
                                         ?Aircraft2.Trajectory)
                               (Feet 400)))))))
...
19 --> (Ra> (Need (!Logic (Error-When
                           (!There-Exists
                            ((?Aircraft1 Aircraft)
                             (?Aircraft2 Aircraft))
                            Such-That
                            (< (Distance (Flight-Plan ?Aircraft1)
                                         (Flight-Plan ?Aircraft2))
                               (Feet 700)))))))
...
20 --> (Ra> (Need (!Logic (Error-When
                           (!There-Exists
                            (?Aircraft1 Aircraft)
                            Such-That
                            (< (Distance ?Aircraft1.Trajectory
                                         (Flight-Plan ?Aircraft1))
                               (Meters 300)))))))
...
```

At this point, the analyst decides to take a break. Before doing this, the analyst tells the RA to strengthen its reasoning power temporarily and reason about what has been specified so far. This is done by means of the think harder command. Because the reasoning of the RA is incomplete, there are often deductions that the RA has not made or contradictions it has not detected. The think harder command does not enable complete deductive reasoning for that is an exponential task, but it does cause more deductions to be made.

```
 21 --> (Ra> (Think-Harder))
 Conflict-# 2 Error-When
   (Not (And (Event-Happens Add-Traj) (Not (Event-Happens Add-Fp))))
 Has-Static-Truth-Value-Is-Tautologous
 Premises:
1. (Need (!Logic (Comes-Before Add-Fp
                               (One-Of Aircraft-Hand-In
                                       Aircraft-Take-Off))))
2. (Need (!Logic (Preceded-By-Strictly Add-Traj
                                       (One-Of Aircraft-Hand-In
                                               Aircraft-Take-Off))))
3. (Need (!Logic (Error-When (And (Event-Happens Add-Traj)
                                  (Not (Event-Happens Add-Fp))))))
...
Resolve conflict?(Y or N) Yes.
Would you like to retract a premise?(Y or N) Yes.
Retract premise # (0 is None)? 2
You-Chose (Need (!Logic (Preceded-By-Strictly
                         Add-Traj
                         (One-Of Aircraft-Hand-In
                                 Aircraft-Take-Off))))
 Resolved-Conflict# 2
 Lost-Value (!Logic
 (Preceded-By-Strictly
  Add-Traj
  (One-Of Aircraft-Hand-In Aircraft-Take-Off)))
  For Aircraft-System.Needs.
```

In this case, the additional deductions made by the RA cause a contradiction to be detected. The RA has determined that the error the analyst specified to be signalled if a trajectory is added to the AMS but no corresponding flight plan exists cannot occur (Command 17). The RA enforces the expectation that all errors can occur. If an event (error) cannot occur, it makes little sense to specify what to do if it does.

It is impossible for a trajectory to be added to the AMS without there already being a corresponding flight plan because an add trajectory event is always preceded by either an aircraft hand in or aircraft take off, and both of these events are preceded by the addition of a flight plan.

Thinking about this, the analyst realizes that the present model does not account for the event of an aircraft entering the observation region without another aircraft system warning the AMS (the hand in event). Hence to resolve the contradiction, the analyst retracts the statement that the add trajectory event is preceded by either an aircraft hand in or an aircraft take off. Then, in command 22 the analyst asserts a corresponding statement that allows for the possibility of unauthorized entry.

```
22 --> (Ra> (Need (!Logic (Preceded-By-Strictly
                           Add-Traj
                           (One-Of Aircraft-Unauthorized-Entry
                                   Aircraft-Hand-In
                                   Aircraft-Take-Off)))))
...
```

Again, the analyst tells the RA to think harder. This time to see if the problem has been corrected and indeed it has.

```
23 --> (Ra> (Think-Harder))
```

Reformulation

In command 24, the analyst decides to tell the RA that the data given by the radar needs to be matched to the aircraft it comes from. This needs to be done because the radar observes many aircraft each time it scans the observation region and does not observe these aircraft in sufficient detail to distinguish their identities. However, the RA thinks that the aircraft are already matched to the observations of the radar since the set of aircraft is the target of the radar system and tracking systems assume that their data and observations are already matched.

This results in a contradiction since the RA enforces an expectation that needs are not statically true: if a proposition is known, it is not necessary for it to be a need, but if the analyst thinks it should be a need, it might be the case that the previous description was incorrect and the proposition does in fact have to be a need.

```
24 --> (Ra> (Need (!Logic (Match (!Set-Of* Aircraft)
                                 (Observation Radar)))))
(!Logic (Match (!Set-Of* Aircraft) (Observation Radar)))
  Is-A-Member-Of Aircraft-System.Needs.
Need (Match (Observation Radar) (!Set-Of* Aircraft))
  Is-Statically-True-In-Current-Description.
...
Premises:
1. (= (Observation Radar) Radar.Observation)
2. (User= (Target Radar) (!Set-Of* Aircraft))
3. (Tracking-System Radar)
Resolve conflict?(Y or N) No.
```

The analyst realizes that there is a need for another level of indirection. An intermediate object, blip, that the radar will track, is created. The system will have to match the blips to the aircraft. Up to this point, the analyst has been using the word aircraft to mean both blip and aircraft. Therefore, in some places the analyst has used aircraft in the place of blip. The `Reformulate` command allows the analyst to selectively replace aircraft with blip throughout the transcript.

```
25 --> (Ra> (Reformulate Aircraft As Blip))
```

Since the aircraft are the actual physical objects the system deals with, the definition of aircraft is not one for blip. Command 2 was correct as originally stated.

```
2. (Define Aircraft :Ako Physical-Object
     :Member-Roles ((The-Trajectory Trajectory)
                    (Flight-Plan Trajectory)))
=Reformulate=>?(Y or N) No.
```

Command 3 on the other hand is changed since the radar does not have the aircraft matched with the observed data, but rather only observes the data and reports it. Another part of the system must match the aircraft with the data of the radar (blips).

```
3. (Define Radar :Tracking-System
     :Roles (:Target (!Set-Of* Aircraft)))
=Reformulate=>?(Y or N) Yes.
```

Command 8 does not change since the beacon of an aircraft reports not only the aircraft's height, but also its identity.

```
8. (Define Beacon :Data-Source
     :Roles (:Error-Value (Feet 100) :From Aircraft))
=Reformulate=>?(Y or N) No.
```

Command 11 does not change since blips do not have flight plans, and the AMS needs the flight plan of an aircraft to make its comparison.

```
11. (Need
     (!Logic
      (Monitors
       Aircraft-Monitoring-System
       (!Image-Of (!Set-Of* Aircraft) Trajectory)
       (!Image-Of (!Set-Of* Aircraft) Flight-Plan))))
=Reformulate=>?(Y or N) No.
```

The other command that needs to be changed is command 18. When the error this command states occurs, two aircraft are too close. This should be signalled as soon as it is known since planes crashing into each other is very undesirable. For this reason, the error should be signalled as soon as the radar gets two data values that are too close, even if they are not matched to aircraft.

```
18. (Need
     (!Logic
      (Error-When
       (!There-Exists
        ((?Aircraft1 Aircraft) (?Aircraft2 Aircraft))
        Such-That
        (< (Distance ?Aircraft1.Trajectory ?Aircraft2.Trajectory)
           (Feet 400))))))
=Reformulate=>?(Y or N) Yes.
```

Commands 19 and 20 cannot change for the same reason as command 11: flight plans are desired and blips do not have flight plans. Command 24 is the command which causes blips to be matched to aircraft. Since the radar now tracks blips, it also does not change.

```
19. (Need (!Logic (Error-When
                   (!There-Exists
                    ((?Aircraft1 Aircraft) (?Aircraft2 Aircraft))
                    Such-That
                    (< (Distance (Flight-Plan ?Aircraft1)
                                 (Flight-Plan ?Aircraft2))
                       (Feet 700))))))
=Reformulate=>?(Y or N) No.
20. (Need (!Logic (Error-When
                   (!There-Exists
                    (?Aircraft1 Aircraft) Such-That
                    (< (Distance ?Aircraft1.Trajectory
                                 (Flight-Plan ?Aircraft1))
                       (Meters 300))))))
=Reformulate=>?(Y or N) No.
24. (Need (!Logic (Match (!Set-Of* Aircraft) (Observation Radar))))
=Reformulate=>?(Y or N) No.
 Retracting-Reasons (3 18)
 Command (Define Radar :Tracking-System
           :Roles (:Target (!Set-Of* Blip)))
 Command (Need (!Logic (Error-When
                        (!There-Exists
                         ((?Aircraft1 Blip) (?Aircraft2 Blip))
                         Such-That
                         (< (Distance ?Aircraft1.Trajectory
                                      ?Aircraft2.Trajectory)
                            (Feet 400))))))
```

```
(!Logic
(Error-When
 (!There-Exists
  ((?Aircraft1 Blip) (?Aircraft2 Blip))
  Such-That
  (< (Distance ?Aircraft1.Trajectory ?Aircraft2.Trajectory)
     (Feet 400))))) Is-A-Member-Of Aircraft-System.Needs.
Lost-Value (!Logic
(Error-When
 (!There-Exists
  ((?Aircraft1 Aircraft) (?Aircraft2 Aircraft))
  Such-That
  (< (Distance ?Aircraft1.Trajectory ?Aircraft2.Trajectory)
     (Feet 400))))) For Aircraft-System.Needs.
Radar.Target Has-Value (!Set-Of* Blip).
```

3 What Was Learned

The RA Can Be Applied in More Than One Domain

The example of requirements acquisition for an air traffic control system shows that the RA can be applied in a domain other than that of library databases. The same disambiguation, contradiction, and incompleteness detection techniques were used in both the library database example and the air traffic control example. While this certainly is not a complete testing of the RA, this example and Reubenstein's legal example argue for the possibility that the RA has wide-range applicability.

Some Clichés Are Shared

While the RA was the same in both the library database and ATC examples, the knowledge provided to the RA (it's cliché library) was not the same in both cases. The union of the cliché libraries used in each example would be able to serve as data for both examples simultaneously. However, this was not done for reasons described below.

Some clichés were shared between the two examples despite the fact that the cliché libraries are different. At the highest level, they shared the same environment, system, and needs clichés. Reubenstein argues that these three kinds of information are a relevant division of the kinds of information used in formalizing, completing, and disambiguating a requirement. Again, this example lends credence to such an argument.

At a lower level, the two scenarios use the same physical object cliché and similar tracking system clichés. While the tracking systems in the two scenarios are not the same, they encompass the same ideas. They are different only because the two creators of the clichés had different ideas about what abstract ideas compose a tracking system. We believe that a hybrid tracking system cliché could be constructed that would encompass the ideas of both authors (Reubenstein and myself) and be able to be disambiguated to particular tracking systems that would be relevant to the individual scenarios. The important ideas of a tracking system tracking an object or set of objects and observing data about the object are the same in both clichés. The differences are that Reubenstein's definition fleshes out more detail because he needed more information to be derived from his cliché and similar roles are given different names.

Disambiguation is Promoted by Combination of Clichés

Another useful thing learned from doing the air traffic control example is that combining clichés in different combinations can create other useful clichés. It is often useful to define a few simple clichés whose combination forms a desired complex

cliché and then think about combining the simple clichés in different combinations. For example, in constructing the limited position cliché (that is a position that is supposed to be in a certain volume), the limited cliché was such a supertype.

After the limited cliché was created, it could be intersected with the position cliché to create the limited position cliché. Also, by considering limited in the context of both the tracking system and error clichés, the direct observation tracking system and limited error clichés were created. A direct observation tracking system is a tracking system that only tracks objects inside a certain volume. Thus, it is limited. A limited error comes from an object known to be inside a certain volume. Thus, the creation of the limited cliché not only allowed definition of the limited position cliché, but also provided the impetus for creating two other clichés: the direct observation tracking system cliché and the limited error cliché. The limited error cliché is particularly useful because the restriction to a particular volume of observations made with error allows conversion of the error from one coordinate system to another.

Combining types is also useful when the RA needs to view an object in two different ways and convert values from one view to another. For example, the distance function might be applied to an object (A) that is viewed as having a polar position and another object (B) that is viewed as having a cartesian position. (This is a specific case of the problem of applying a function to two objects which can be compared only after one undergoes some conversion.) It is perfectly reasonable for the distance function to be expected to convert the polar coordinates of A to the relevant cartesian coordinates (or vice versa), but implementationally it is easier to make A an instance of the polar cartesian position cliché (where the conversion will occur automatically), and then just apply the polar distance function to A and B (the RA will do this automatically if the relevant cliché definitions are present). This has the advantage that the distance function is simpler. Also, the intersection types are more widely applicable since the need for them may arise in other situations. For example, if the user were to specify the range and latitude of an object then the object should be an instance of both the polar position and cartesian position clichés.

4 Contradiction Summarization

The principle difficulty encountered in this example was deciphering contradictions the RA detected. In this section, some solutions to this problem are suggested.

At the present time, only the highest level clause and the underlying premises of a contradiction in the RA are displayed in abbreviation mode. Here, a clause is a disjunction of nodes (some of which are designated as positive nodes, and others that are designated as negative nodes). A contradiction occurs when all of the positive nodes in a clause are false and all of the negative nodes are true. A node is an instance of a data abstraction in the RA's three valued logic. It can be true, false, or unknown. If all of the nodes in a clause except one (N) are as they would be in a contradictory clause, and N is unknown, then N is given a truth value (true or false) so that the clause is not contradictory (i.e., true if N is a positive node, and false if N is a negative node.). This clause is the support of N. A node is a premise if it has no support. (i.e., the RA explicitly set its truth value.)

While not in abbreviation mode, the RA shows a linearization of the whole support structure of a contradiction. This is often quite lengthy. Since the RA only displays contradictions totally or very briefly, determining the best way to correct one can be challenging. The RA installs many clauses and asserts many nodes as premises when a requirement is being defined. Also, many clauses are used in the RA's deduction process for most nodes involved in a contradiction. I developed a tool to display these graphically in the form of a tree that shows the relevant dependencies about which nodes support which other nodes. The display tool has been useful in other phases of the Programmer's Apprentice Project. Incorporating it into the RA's contradiction recovery mechanism would make deciphering contradictions much easier. A tree is much more effective in displaying the contradiction than a linearization since the reasoning involved is seldom linear.

However, the trees produced in showing a contradiction can often be quite large. Displaying the complete deduction process performed by the RA is not optimal since such a display would take too long to decipher. A middle ground between giving the analyst too little information to understand the contradiction (as is presently the case in the RA) and giving so much information that it is indecipherable (as would be the case if the complete tree generating tool were incorporated into the RA) needs to be found.

In the display of a contradiction as a tree, the nodes of the tree are printed representations of nodes in the RA and their truth values. The arcs represent the dependencies given by the clauses. The children of a node (N) are the nodes in the clause (C) that supports N (except N itself). Also, the documentation string associated with C is displayed in the node describing N.

As said above, these trees grow quite large quite fast. However, they can be pruned somewhat. Two methods that have been experimented with in attempting to

reduce the size of these trees without significantly reducing the amount of information they contain are: removing nodes and removing all of the nodes below a certain node (truncation). Obviously, the nodes on which this should be done must be chosen carefully.

The first kind of node that can be removed is one that contains information internal to the RA. This is a fairly obvious thing to do and is relatively uninteresting. Also, the nodes supporting these nodes can often be removed (truncation). Care needs to be taken in doing this because important information can be lost: informative nodes may support the node below which we are truncating.

A more interesting type of node that can be removed is a node that is the conjunction of two other nodes. For example, if the node (And A B) is supported by A and B (i.e., the conjunction is true because both of the disjuncts are true.), then the node (And A B) can be removed because it is obvious. If P is the parent of (And A B) in the original tree, P will be supported directly by A and B as well as any nodes it was supported by before (except (And A B)). Very little information is lost to the analyst because conjunction is something that people do automatically.

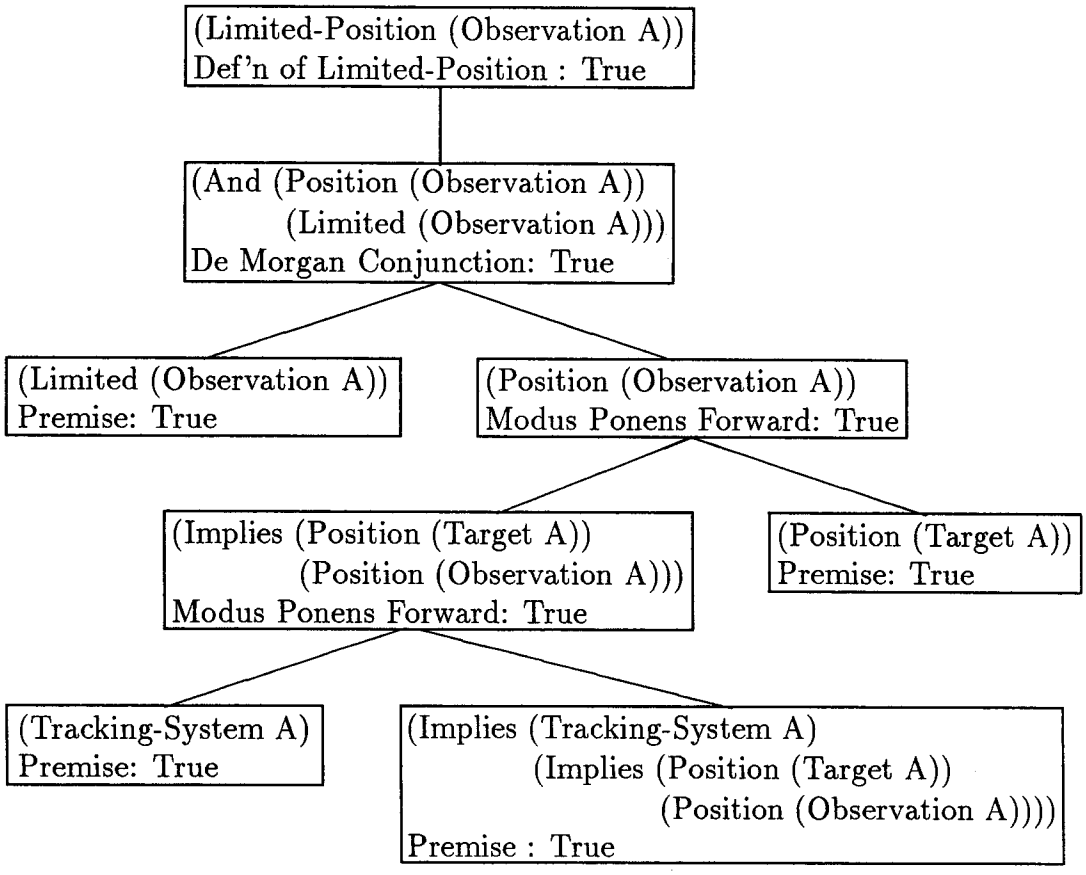
Another type of node that can be removed is one of the form (Implies A B) when B is supported by it and A. The node (Implies A B) gives the information that B can be inferred from A, but that information is implicit in the tree since A supports B. However, the nodes that support (Implies A B) need to be annotated so that the analyst knows they support the implication node and not B directly. This is done by putting the documentation of the clause that supports the node (Implies A B) on those nodes (i.e., the documentation that is normally in the node containing (Implies A B)). This is an example of a general heuristic: when a node (N) is removed, put the documentation string of N (the documentation on the clause supporting N and any other documentation pushed to N by previous pruning) on the nodes supporting N.

A refinement of the pruning process is to leave premises in the tree. Premises are particularly important in the RA because they represent the underlying assumptions the analyst has made that resulted in a contradiction. It is only by retracting or changing them that the analyst can cause the contradiction to be resolved. For this reason, premises are sometimes left in the tree even if another rule specifies that they should be removed.

Figure 1 gives an example of an unpruned tree and Figure 2 shows a pruned proof of the same node. The node `(And (Position (Observation A)) (Limited (Observation A)))` has been removed because its truth follows from both conjuncts. The Implies nodes have also been removed. The premise was removed because it is part of the definition of tracking system. Were it a user asserted premise, it would not have been removed.

The documentation from `(Implies (Position (Target A)) (Position (Observation A)))` was transferred to `(Tracking-System A)` when the first was re-

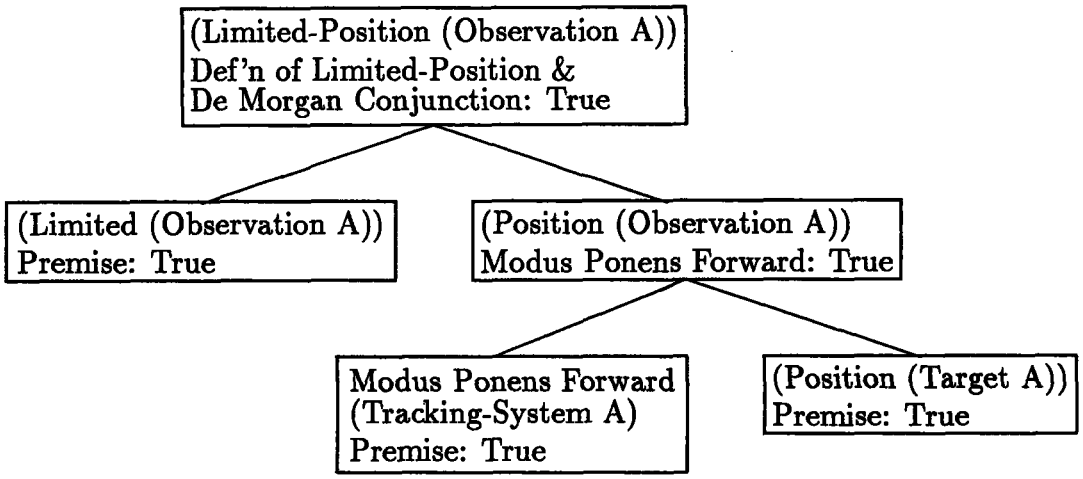
Figure 1: Unpruned Proof of (Limited-Position (Observation A)).



moved. When the And node was removed, it's documentation was also moved to both of its children, but since after pruning, both children of `(Limited-Position (Observation A))` shared common documentation, it was moved to the parent so that it wouldn't be printed multiple times.

It is not always the case that too much information is displayed in the trees produced by the RA. It is also possible that these trees contain too little information even before pruning). This occurs because the RA has mechanisms that produce "resolution" clauses. A resolution clause is one the RA knows to be true because of other clauses. Typically, theses clauses are produced when a contradiction occurs within the RA and the contradiction is resolved. When a contradiction results, the RA has proven that the truth values of the underlying premises of the contradictory clause cannot all occur simultaneously. Thus, a clause can be installed that has as positive nodes all of the underlying premises that are presently false, and as negative

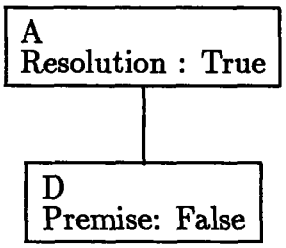
Figure 2: Pruned Proof of (Limited-Position (Observation A)).



nodes, those that are presently true. This means that when the proof of a node supported by this clause is displayed, the analyst will only see the underlying premises. This is often not enough information for the analyst to reasonably determine what is wrong. For this reason, the RA caches enough information at the time the new clause is installed so that a proof by cases can be displayed to the analyst.

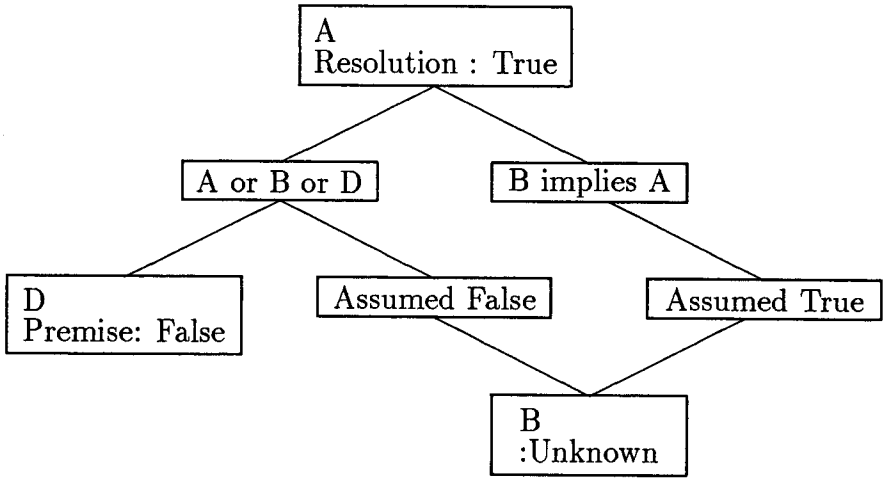
A simple example is the best way to illustrate this. Suppose that the RA has two clauses: C1 with positive nodes A, B, and D and documentation `A or B or D`, and C2 with positive node A and negative node B and documentation `B implies A`. Logically, these clauses can be thought of as $A \vee B \vee D$ and $A \vee \neg B$ respectively. Now, if the analyst were to assert A to be false and D to be false then the RA could do one of two things. Either it could deduce B to be true from C1, and C2 would become contradictory, or it could deduce B to be false from C2, and C1 would become contradictory. Let us assume that the first occurs and that the analyst resolves the contradiction by retracting A. Now the RA installs a clause C3 with A and D as positive nodes and no negative nodes (A and D were the premises underlying the contradiction and were both false at the time). This has the effect of making A true.

Figure 3: Original proof of A.



Under the original scheme of showing proofs, the only reason given for the node A being true is that D is false and the documentation on C3. This documentation is typically "resolution" and gives the analyst no more information than knowing that A is true based on other clauses in the database (see Figure 3). What the analyst really wants to see here is a proof by cases. That is, if B is true, then C2 forces A to be true (since D is false) and if B is false, then C1 forces A to be true (see Figure 4). It might be the case that B is true, false, or unknown, but whatever its truth value, the proof mentioned above is valid and gives the analyst more information. The nodes whose truth value doesn't matter are often referred to as split nodes. In large proofs there can be many split nodes and so it is also useful to allow the analyst to decide which cases are interesting and only display those. While in general there is not a worst case exponential explosion of the number of cases involved in a proof by cases, there can be many cases and allowing them to be pruned is useful. Having a tool to automatically decide which of the cases is interesting would also be useful.

Figure 4: Proof by cases of A.



5 References

[1] C. Rich, "The Layered Architecture of a System for Reasoning about Programs," *Proc. 9th Int. Joint Conference on Artificial Intelligence*, pp. 540-546, August 1985.

[2] C. Rich and R.C. Waters "The Programmer's Apprentice: A Research Overview," *IEEE Computer*, 21(11):10–25, November 1988.

[3] C. Rich, R.C. Waters, and H.B. Reubenstein, "Toward a Requirements Apprentice," in *Proc. 4th Int. Workshop on Software Specification and Design*, pp. 79-86, IEEE Computer Society Press, Washington DC, April 1987.

[4] H.B. Reubenstein and R.C. Waters, "The Requirements Apprentice: An Initial Scenario," In *Proc. 5th Int. Workshop on Software Specification and Design*, IEEE Computer Society Press, Washington DC, May 1989.

[5] H.B. Reubenstein and R.C. Waters, "The Requirements Apprentice: Automated Assistance for Requirements Acquisition," 1990.

[6] H.B. Reubenstein, *Automated Acquisition of Evolving Informal Descriptions*, Ph.D. Thesis, MIT/AI/TR-1205, MIT Artificial Intelligence Lab., (to appear, June 1990).

Many thanks to Howard Reubenstein for his many useful comments about creating cliché libraries and how exactly the RA does what it does (and doesn't). Also, thanks go to Richard Waters for keeping me on the right "higher" level track and Charles Rich for arguing with me about how contradiction summarization/expansion should be done. Finally, thanks to Yang Meng Tan and Linda Wills for putting up with me in their office. Thanks to all for proofreading. Any errors that remain are mine.