Shruti Chandrasekhar
6.871
Final Project

The Constitution Builder

This system helps constitution frames decide on and structure certain aspects of constitutions. More specifically, it helps the constitution designer decide if federalism is the appropriate choice for a given country based on the country's specifics.

Background

Constitutions are not an exact science but are founded in logic and reason. Government structure as dictated by a constitution shapes the life of the democracy. It is important to construct this document right in order to build a great nation. Constitutions specify various aspects of a government ranging from the right kind of electoral system to powers vested in the head of state. All these specifications are founded on idealistic political thought but built primarily on knowledge gained by reviewing performance of other governments over the course of time. This wealth of knowledge helps constitutional engineers learn from the problems governments in similar countries have faced and preempt and prevent them. A field so steeped in knowledge is ideal for knowledge engineers to transform and facilitate so as to aid development in that subject.

The wealth of knowledge that I refer to is better described through the following example. Let us assume in this day and age of world democratization we are faced with a country in the need for a new government. Given the ability to set up virtually any kind of government what would we base our choices on? With the goal of creating an enduring stable democracy we would start out from the current situation of the given country. The country in question is a fairly large country both in area and population, has a history of oppressive dictatorships, predominantly follows one religion, has three significant ethnic groups with the largest having a significantly larger population percentage than the following two and is in need of considerable indigenous development, based on these factors experts would probably suggest a federal parliamentary system with proportional representation, double balloting and a controlled economy. These choices are largely based on other countries with analogous backgrounds that have made similar decisions which have proved successful over the years. This is the knowledge based rationale that constitutional design is built upon.

With this background in mind, we decided work on this area of political science for our final project. However due to the breadth of this field we decided to focus our attention solely to one aspect of constitutional design that most framers strongly consider while setting up a government – the issue of federalism. Federalism, in the words of William H. Riker – one of the great minds of modern day political science, *is a political organization in which the activities of government are divided between regional governments and a central government in such a way that each kind of government has some activities on which it makes final decisions*[1].

Federalism is commonly thought of as the division of power between a central governing body and geographically distributed local provincial governments. It is debated that federalism is more akin to sharing power than dividing power between the center and the states[2] which is one of the main differences between federalism and majoritarian governments. Another aspect that distinguishes federal powers is the power distribution is much less one-sided than that of a majoritarian government. In a majoritarian government, the power is highly biased towards the center with the states getting little, if any, real power. Another reason why federalism seems so appealing is because it seems to create a microcosm of the country in its government giving people the assurance of fair representation. Division of government increases efficiency, brings the government closer to home, reduces running costs, encourages, empowers and enables more citizens to participate in the state's decision making processes, and helps train future leaders for the national arena by letting them make their mistakes on a smaller scale. In all, a number of very attractive reasons make federalism much desired.

However there are many circumstances under which these benefits may actually harm a country and lead to federalism not being the best option, or even the worst option. So when do we choose federalism over all else? Federalism is based on the concept of grassroot governments that build on to create a larger tree. However trying to chalk out sub-sections of countries not big enough to be grass themselves would make federalism a joke. Secondly, trying to create geographic divisions when the population and resources are unevenly distributed would make sustenance of the federal state an ordeal. Finally the crucial issue when federalism is almost always a necessity is the existence of diversity. A diverse society needs to be handled differently from a homogenous one. All divisions and minorities need to be recognized and appropriately incorporated into the government to build a cohesive stable state. When federalism is incorporated in a country divided by religion or ethnicity or any other such societal cleavage, it gives each community a sense of self-governance while still maintaining the country as a unitary whole. This has historically been the primary reason for adopting a federal form of government.

Federalism has come to be in many ways. Two often occurring forms of federal states are coming-together federalism and holding-together

[1] Riker, William H. 1975. "Federalism," in Fred Greenstein and Nelson W. Polsby, eds., Handbook of Political Science: Governmental Institutions and Processes. Addison-Wesley. For example, Riker's (1987:101) definition is often used, despite its ambiguity: "a political organization in which the activities of government are divided between regional governments and a central government in such a way that each kind of government has some activities on which it makes final decisions."

[2] Arend Lijphart, Patterns of Democracy: Government Forms and Performance in Thirty-Six Countries. Chapter 10 – Division of power: The Federal-Unitary and Centralized-Decentralized Contrasts. Yale.

federalism[3]. Some countries are formed by neighboring states coming together to form a union, as in the case of the United States or Canada, for better utilization of resources and people and a better life. In such cases federal structures work best. Some countries choose federalism when they have significant differences amongst their geographically dispersed population but some essential similarities and hence need a glue to hold them together as a nation as was the case in India and Belgium. In all there are a number of reasons when federalism should be the obvious or the not-so-obvious choice – our project helps identify the situations when federalism would or would not work.

Problem Solving Paradigm

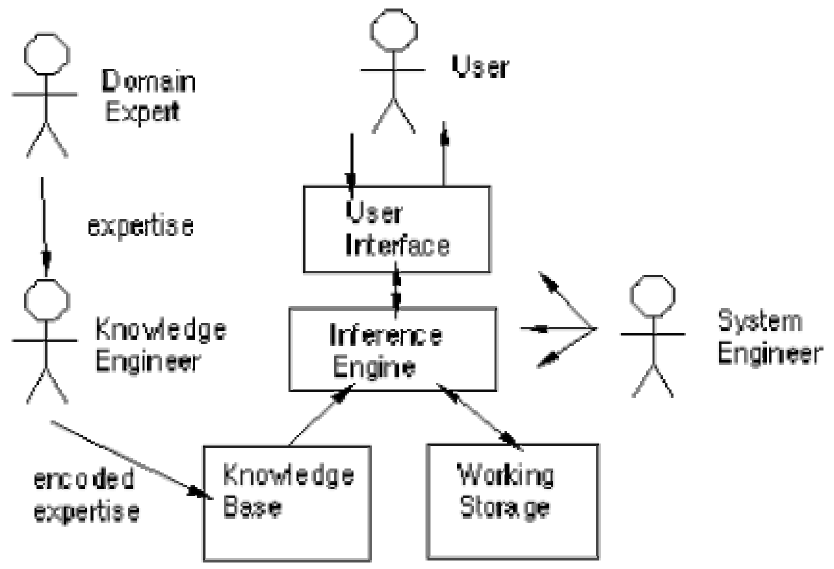
Due to the logical flow of knowledge and reasoning in this area of study we decided to approach the problem using a flow-chart model. By flow-chart model I mean an attribute-corollary setup. The system looks at specifics of the country that its knowledge database considers crucial to the question of federalism. It queries for these specifics using the knowledge framework laid out, outlines the thread of logic dictated by the system and then extrapolates whether federalism would work or not. The more we thought about such a system, the more appropriate such a characteristic-inference choice seemed. Hence rules seemed a very logical follow-up. Due to this multi-layered *attribute–corollary – attribute clarification – inference modification* – like treatment of the problem, rules would be the optimal choice.

System Setup

Expert systems have a number of major system components and interface with individuals in various roles. These are illustrated in figure below. A knowledge engineer (in this case us) capture the expertise from a Domain Expert and encode it into a knowledge base. The system engineer (here, again it is us, using information derived from Prof.Davis, Jacob Eisenstein and the Joshua reference manual) create a system that acts as an inference engine, working storage and a user interface. In detail the major components are:

- Knowledge base - a declarative representation of the expertise, often in IF THEN rules;
- Working storage - the data which is specific to a problem being solved;
- Inference engine - the code at the core of the system which derives recommendations from the knowledge base and problem-specific data in working storage;
- User interface - the code that controls the dialog between the user and the system.

[3] Stepan, Alfred (1999) "Towards a New Comparative Analyisis of Democracy and Federalism: Demos Constraining and Demos Enabling Federations" Mimeo, Oxford.



To understand expert system design, it is also necessary to understand the major roles of individuals who interact with the system. These are the:

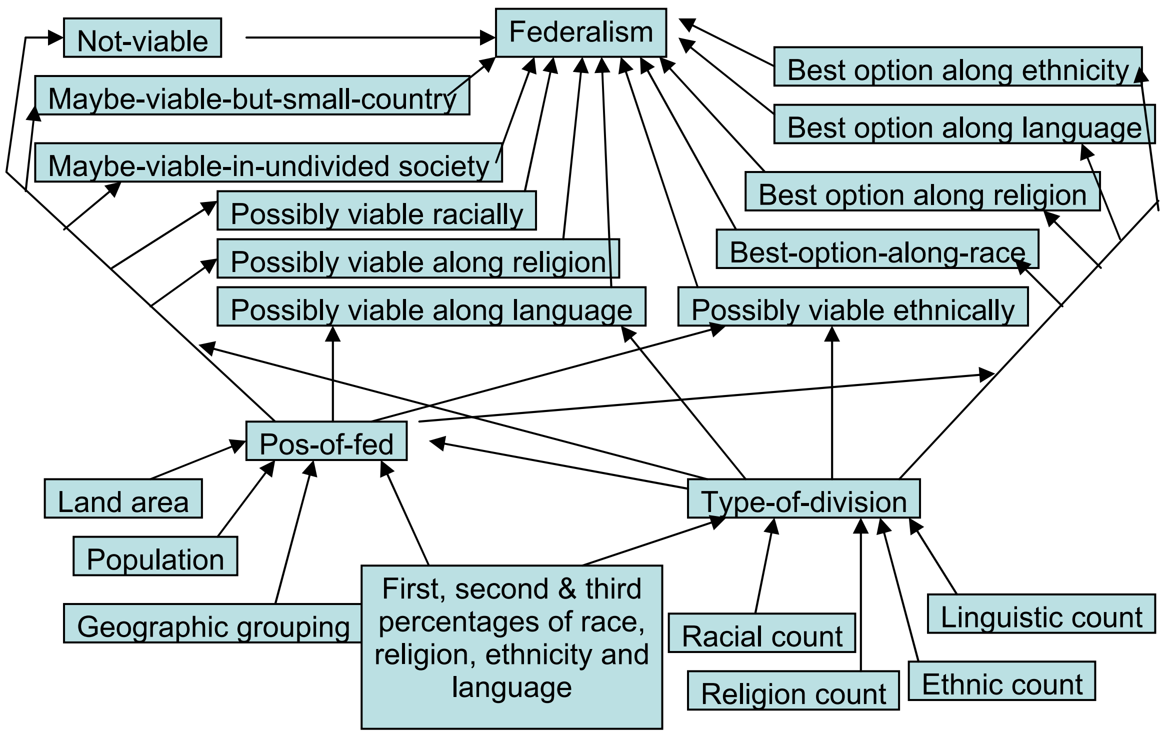
- Domain expert - the individual or individuals who currently are experts solving the problems the system is intended to solve;
- Knowledge engineer - the individual who encodes the expert's knowledge in a declarative form that can be used by the expert system;
- User - the individual who will be consulting with the system to get advice which would have been provided by the expert.

Knowledge Base

The system is built upon a number of cumulative rules. These rules are fairly exhaustive. The rules in themselves contain the fundamental knowledge about federalism. The rules establish what the system 'knows' from what population range best fits a federal state to how appropriate spatial distribution of diversity would be towards federalism. The rules embody the core of this expert system. The rules are written in LISP in Joshua. It is organized as layered memory using backward chaining. Backward chaining was adopted here since a backward chaining system works backward from the initial goal of trying to answer a question posed by the user[4], which in this case is *if federalism would work towards creating a stable enduring democracy for the country in question*. In answering this question, the system spawns subgoals to answer further questions posed by the conditions of the rules defined. Such systems do represent their goals explicitly as they have to in order to return up a level. In our system we are trying to pick the best choice from many enumerated possibilities. Since this is in-between an identification problem and a diagnostic system and is predominantly based on goal driven reasoning we decided to adopt backward

[4] Jocelyn Paine, Lecture notes on rule-based systems, 1996 http://www.j-paine.org/students/lectures/lect3/node1.html

chaining[5]. The rules are included in the appendix. The rules are broadly classified into rules dealing with size, rules assigning type of division, rules calculating possibility, rules about geographical distribution and finally rules assessing kind of federalism or none at all.


Not-viable
Federalism
Maybe-viable-but-small-country
Best option along ethnicity
Best option along language
Maybe-viable-in-undivided society
Best option along religion
Possibly viable racially
Best-option-along-race
Possibly viable along religion
Possibly viable along language
Possibly viable ethnically
Pos-of-fed
Land area
Type-of-division
Population
Geographic grouping
First, second & third percentages of race, religion, ethnicity and language
Racial count
Linguistic count
Religion count
Ethnic count

Each rule is expressed in terms of an *if* part (the premise) and a *then* part (the conclusion). Each clause in a rule is expressed in terms of an attribute, object and its value. For instance, consider rule 4,

```
if    population >= 10,000,000 and
      land_area >= 100,000
then  pos_of_fed = 0.5
```

In English, this says –

If the population of the country is larger that ten million and
If the land area of the country is larger that 100,000 sq km
Then the numeric factor measuring possibility of federalism is 0.5

The progression of ideas when asked if federalism is appropriate for country 'A' is – does A have significant societal cleavages? How strong are these cleavages? Are they geographically distinct as well? Is the population large enough to sustain such a cleavage?. Similarly, the progression of rules when asked *(ask [federalism IRAQ ?x] #'print-answer-with-certainty)* follows suit.

[5] Amzi.com – Expert systems in prolog

What is A's number of races?

What is A's most prevalent race's population percentage (enter 0.3 for 30%)?

➔ *if the checks validate the type_of_division is assigned to none.*

Thus the rules try to replicate the mind of an expert.

Inference Engine and User Interface

The inference engine takes in a number of inputs regarding the situation of the country in question. The questions range from size of the country's population to the various kinds of diversities located in the country. The output consists of the decision whether federalism is a fit or not, along with how appropriate a fit it might be. It also outputs how the new federal structure to be created (if that be the case) should be laid out – as in should the provincial demarcations be based on race or religion or language as so forth. If asked for an explanation, the system would reason with the user as to how and why it arrived at the conclusion it did.

The inputs and outputs defined in the inference engine are as follows –

Inputs		
Variable	Description	Possible Values
Population		All numeric values
Land Area (in sq.km)		
Racial Count	Number of races living in the country	
Religion Count	Number of religions practiced by the people	
Ethnic Count	Number of ethnicities occupying the country	
Linguistic Count	Number of languages spoken in the country	
first_religion_p, first_racial_p, second_ethnic_p, …	Population Percentages of First, Second and Third highest in race, religion, ethnicity and language	Decimal value in the range of 0 to 1
Geographic grouping	States what kind of subgroups occupy the different regions of the country	None / racial / religion / ethnic / linguistic
Output		
Variable	Description	Possible Values
Type of Division	The most prominent societal cleavage	None / racial / religion / ethnic / linguistic
Possibility of Federalism	A degree measure of how appropriate a fit federalism would be for the country	Numeric value between 0 and 1

Federalism	The final answer on the decision of federalism or not	not-viable / maybe-viable-in-undivided-society / maybe-viable-but-country-very-small / possibly-viable-racially / possibly-viable-along-religion / possibly-viable-ethnically / possibly-viable-along-language / best-option-along-race / best-option-along-religion / best-option-along-ethnicity / best-option-along-language))

The thread of logic starts out from an ask query issued by the user. When the user asks a question on how suitable federalism is the inference engine looks are the two main factors that help resolve this query, namely societal cleavages (type_of_division) and a numeric factor that encompasses all other aspects (pos_of_fed). In order to assess the extent of societal cleavages, it steps into to discover data about the given case. It queries information on the population breakup along lines of race, religion, ethnicity and language. On obtaining this information as desired, it analyses and concludes on the most prominent cleavage. It then tries to incorporate this information into its calculations of the numeric factor. It starts its calculation on this factor but sizing up the size of the country in order to eliminate very small country where federalism might be more an expense than an aid. It then progresses using the information on the population percentage breakup of the prominent social cleavage to augment this numeric factor. Finally it tries to obtain information on how this social divide is located geographically as in are people on one side of this divide concentrated in location and arrives at a conclusive value. During all these calculations it does not obliterate the other cleavages, in fact it goes further to calculate the effect of federalism due to those cleavages as well. It finally outputs an answer on the appropriateness of federalism for that country ranging from not viable to best option available. It also states what the major divide was, if any, and if the country's provinces should be delineated based on this divide. If more than one divide exist it suggests federalism on both divides if possible but includes a factor (certainty) a higher value of which indicates the option that would be more appropriate.

The user interface consists of a series of queries as it traces through the rules. It shoots queries out to the user to elicit information about the country in question. The question backtrack through the rules and usually try to establish the type of division and a value for the possibility of federalism in order to evaluate the decision on federalism as a fit for the country. The user interface is very basic and error checking techniques are not very robust. But if instructions are followed through the interface is a simple sequential interview-like setup.

Working Storage

The working storage is the data stored in the variables specified in Joshua. This is the information obtained as a result of the queries. For example this would be the actual population of the country in question or the percentage of people who follow Christianity or the number of races that live in that country and so forth. This information is input by the user while running the program.

Domain Expert

As a knowledge-based application system for constitutional design, this topic is far removed from the world of computer science and artificial intelligence. Due to this aspect a Domain Expert was crucial to the structure of this system. We worked with Prof.Cindy Skach. Prof.Skach is a Professor of Government at Harvard University. Her field of research is comparative constitutional law, constitutional engineering, and the theory and practice of democracy. Due to specialization in this field she was invaluable to the construction of our system. Her time constraints forced us to work occasionally with her teaching fellow, Rosalind Dixon. Rosalind is a trained constitutional lawyer, who practiced law for a number of years before enrolling at Harvard Law for her S.J.D. Combined with the knowledge of these experienced professionals we also consulted a number of prominent books in the field ranging from Lipjhart's Patterns of Democracy to Sartori's Comparative Constitutional Design. A complete list of books has been included in the Appendix.

Example Run of the System

Here is an example run of the program.

Let us consider the example of Iraq. Iraq a country of approximately 26 million people, spread over a land mass of 500,000 sq. km has had a tumultuous political past. Periodic forced division from Britain's policies of divide and conquer to Saddam Hussein's Kurdish-Sunni infusions to current Shiite power, societal divisions have shaped Iraq's past. A complete federal structure would prevent the minority Kurds and Sunni from being active participants in the developing government. Given information about Iraq experts would suggest a semi-federal structure as this would give the Sunnis, Shiites and Kurds significant power at the provincial levels and still create a cohesive state.

Our system suggests the following -

(ask [federalism IRAQ ?x] #'print-answer-with-certainty)

```
What is IRAQ's number of races: 1
What is IRAQ's most prevalent races population percentage
(enter 0.3 for 30%): 1
What is IRAQ's number of religions practiced: 1
What is IRAQ's most followed religions population percentage
(enter 0.3 for 30%): 1
What is IRAQ's number of living ethnicities: 3
What is IRAQ's most populous ethnicitys population percentage
(enter 0.3 for 30%): 0.65
What is IRAQ's number of spoken languages: 1
What is IRAQ's most spoken languages population percentage
(enter 0.3 for 30%): 1
What is IRAQ's population: 25,000,000
What is IRAQ's land area: 500,000
What is IRAQ's second most populous ethnicity's population
percentage (enter 0.3 for 30%): 0.21
What is IRAQ's geographical grouping along the lines of
community (none/ethnic/religion/race/linguistic): ethnic
[FEDERALISM IRAQ BEST-OPTION-ALONG-ETHNICITY] 0.5547459
[FEDERALISM IRAQ POSSIBLY-VIABLE-ETHNICALLY] 0.45
[FEDERALISM IRAQ MAYBE-VIABLE-IN-UNDIVIDED-SOCIETY] 0.00001
```

System Extent

The original intent was to setup up a constitution builder – a system that would take in characteristic of a country starting with population and land area all the way through to past dictators, military regimes, minority percentages and so forth and churn the information through the system and produce a bullet point skeleton of the main aspects of the optimal government such as electoral systems, executive (parliamentary or presidential), courts, legislature, etc. However as such a mammoth task would better suit a master's thesis than a final term project, following Prof.Davis' advice we decided to focus on one issue, namely federalism.

This system can handle all sorts of countries in terms of population, diversity, geographics, land area etc. Any possible combination of these attributes is well handled by the system. For instance, if we had an island country that had no diversity with a fairly small homogeneous population, the system would advice against federalism as would an expert. Or if we had a large country, where different corners of the country spoke different languages the system would suggest federalism along linguistics as would an expert. If we had more complex cases for example, a country with three strong ethnic groups that follow two religions where the people are geographically concentrated on ethnicities, the system would suggest federalism along ethnic lines. Hence under most cases the system does replicate the expert.

However there are cases where the system's recommendations are not always perfect. For instance in most small countries federalism isn't a viable option, however there do exist some anomalies such as Micronesia and St.Kitts – both small island states that are federal. If the exact information of these states were input into the system, federalism may not have been suggested. But history tells us that it worked. So the system is not always a hundred percent accurate. Or lets changes the third example cited above slightly, the country has three strong ethnic groups that follow two religions but is geographically concentrated on religion. Our system stipulates a minimum of three prominent subgroups in a social group to suggest federalism on those lines. In this case an expert might have suggested federalism along religion but our system would have still suggested federalism along ethnicity as it deems only two prominent religions as insufficient for a federal setup. Hence there are specific cases where a potentially federal state could fall through the system so as to cause the system to suggest a non-federal government.

A possible way of handling such problems would be use examples as references in the system. If the system feels that federalism would work it should be able to pull out an example from its repository to support its suggestion. Such an extended example-case system would be possible using frames. Trying to rework the system around frames and rules would produce the known best suggestion for anomalies as well.

System Performance

The system captured the train of thought of the expert perfectly as rules are a logical, fairly sequential stream of thought. Backward chaining worked well with the goal-oriented nature of the problem. The if-then style of knowledge encapsulation serves the reasoning nature of constitutional framing substantially well. However Joshua did prove redundant on occasion. In our rules, the diversity rules are nearly identical for the different diversities however Joshua does not permit a generic rule that would serve as an abstraction for all the other rules. In Joshua we had to write each rule over again for each diversity. Secondly Joshua also limited our ability to augment the numeric federalism factor as a part of our rule consequence. This forced us to write six rules where one would have served the purpose in other languages. For instance rules 20-43 are basically 2 rules repeated 3 different ways for 4 different diversities. This could be made more efficient if Joshua handled a more generic rule form.

Lessons learnt, tasks done and knowledge gained – The conclusion

The most important lesson I learned from this is even a small aspect of a given field can be significantly complicated. When we set out to capture federalism I thought it would be a fairly simple task, but I was wrong. Not only can be it be intricately complex but also thoroughly fascinating. It was a neat experience to see how my professor’s knowledge could be encapsulated to some extent. I did notice how frames would have made this project that much more effective and efficient since examples would help infer a better more accurate result along with producing a more compelling argument for or against a suggestion. If I were to work on this task further I would try to incorporate examples in either using frames or some other framework. I would also try to make it more specific in its analysis and try to extend it to different aspects of a constitution. In all, I hope this system helps constitutional framers make the right choices on federalism.

Appendix
Rules

Rules dealing with country size

```
1]    if     population < 10,000,000
             land_area < 100,000
      then  possibility_of_federalism = 0.2

2]    if     population < 10,000,000
             land_area >= 100,000
      then  possibility_of_federalism =  0.35

3]    if     population >= 10,000,000
             land_area < 100,000
      then  possibility_of_federalism = 0.35

4]    if     population >= 10,000,000
             land_area >= 100,000
      then  possibility_of_federalism = 0.5
```

Rules assigning type of division.

Rules about racial diversity

```
5]    if     racial_count > 2 and
             First_racial_percentage < 0.9
      then   type_of_division = racial

6]    if     racial_count <= 2 and
      then   type_of_division = none
```

Rules about religious diversity

```
7]    if     religion_count > 2 and
             first_religion_percentage < 0.9
      then   type_of_division = religion

8]    if     religion_count > 2 and
             first_religion_percentage < 0.9 and
             type_of_religion = racial
             first_religion_division < first_racial_division
      then   type_of_division = religion
```

Rules about ethnic/identity diversity

```
9]    if     ethnic_count > 2 and
             first_ethnic_percentage < 0.9 and
      then   type_of_division = ethnic
```

```
10]   if     ethnic_count > 2 and
             first_ethnic_percentage < 0.9 and
             type_of_division = racial and
             first_ethnic_division < first_racial_division
      then   type_of_division = ethnic

11]   if     ethnic_count > 2 and
             first_ethnic_percentage < 0.9 and
             type_of_division = religion and
             first_ethnic_division < first_religion_division
      then   type_of_division = ethnic
```

Rules about linguistic diversity

```
12]   if     ling_count > 2 and
             first_ling_percentage < 0.9 and
      then   type_of_division = linguistic

13]   if     ling_count > 2 and
             first_ling_percentage < 0.9 and
             type_of_division = racial and
             first_ling_division < first_racial_division
      then   type_of_division = linguistic

14]   if     ling_count > 2 and
             first_ling_percentage < 0.9 and
             type_of_division = religion and
             first_ling_division < first_religion_division
      then   type_of_division = linguistic

15]   if     ling_count > 2 and
             first_ling_percentage < 0.9 and
             type_of_division = ethnic and
             first_ling_division < first_ethnic_division
      then   type_of_division = linguistic

16]   if     (ling_count < 2 and
             Religion_count < 2 and
             Racial_count < 2 and
             Ethnic_count < 2) or
             (first_ling_percentage > 0.9 and
             First_religion_percentage > 0.9 and
             First_racial_percentage > 0.9 and
             First_ethnic_percentage > 0.9)
      Then   type_of_division = none
```

Rules calculating possibility

```
 17]  if    type-of-division = none
      then  first-p = 1

18]   if    type-of-division = none
      then  second-p = 0

19]   if    type-of-division = none
      then  third-p = 0

20]   if    type-of-division = racial and
            first-racial-p <= 0.5 and
            second-racial-p >= 0.3 and
            pos-of-fed = 0.5
      then  pos-of-fed = 0.8

21]   if    type-of-division = racial and
            first-racial-p <= 0.5 and
            second-racial-p >= 0.3 and
            pos-of-fed = 0.35
      then  pos-of-fed = 0.65

22]   if    type-of-division = racial and
            first-racial-p <= 0.5 and
            second-racial-p >= 0.3 and
            pos-of-fed = 0.2
      then  pos-of-fed = 0.5

23]   if    type-of-division = religion and
            first-religion-p <= 0.5 and
            second-religion-p >= 0.3 and
            pos-of-fed = 0.5
      then  pos-of-fed = 0.8

24]   if    type-of-division = religion and
            first-religion-p <= 0.5 and
            second-religion-p >= 0.3 and
            pos-of-fed = 0.35
      then  pos-of-fed = 0.65

25]   if    type-of-division = religion and
            first-religion-p <= 0.5 and
            second-religion-p >= 0.3 and
            pos-of-fed = 0.2
      then  pos-of-fed = 0.5
```

```
26]   if      type-of-division = ethnic and
              first-ethnic-p <= 0.5 and
              second-ethnic-p >= 0.3 and
              pos-of-fed = 0.5
      then    pos-of-fed = 0.8

27]   if      type-of-division = ethnic and
              first-ethnic-p <= 0.5 and
              second-ethnic-p >= 0.3 and
              pos-of-fed = 0.35
      then    pos-of-fed = 0.65

28]   if      type-of-division = ethnic and
              first-ethnic-p <= 0.5 and
              second-ethnic-p >= 0.3 and
              pos-of-fed = 0.2
      then    pos-of-fed = 0.5

29]   if      type-of-division = linguistic and
              first-linguistic-p <= 0.5 and
              second-linguistic-p >= 0.3 and
              pos-of-fed = 0.5
      then    pos-of-fed = 0.8

30]   if      type-of-division = linguistic and
              first-linguistic-p <= 0.5 and
              second-linguistic-p >= 0.3 and
              pos-of-fed = 0.35
      then    pos-of-fed = 0.65

31]   if      type-of-division = linguistic and
              first-linguistic-p <= 0.5 and
              second-linguistic-p >= 0.3 and
              pos-of-fed = 0.2
      then    pos-of-fed = 0.5

32]   if      type-of-division = racial and
              first-racial-p >= 0.6 and
              second-racial-p >= 0.2 and
              pos-of-fed = 0.5
      then    pos-of-fed = 0.7

33]   if      type-of-division = racial and
              first-racial-p >= 0.6 and
              second-racial-p >= 0.2 and
              pos-of-fed = 0.35
      then    pos-of-fed = 0.55
```

```
34]   if     type-of-division = racial and
             first-racial-p >= 0.6 and
             second-racial-p >= 0.2 and
             pos-of-fed = 0.2
      then   pos-of-fed = 0.4

35]   if     type-of-division = religion and
             first-religion-p >= 0.6 and
             second-religion-p >= 0.2 and
             pos-of-fed = 0.5
      then   pos-of-fed = 0.7

36]   if     type-of-division = religion and
             first-religion-p >= 0.6 and
             second-religion-p >= 0.2 and
             pos-of-fed = 0.35
      then   pos-of-fed = 0.55

37]   if     type-of-division = religion and
             first-religion-p >= 0.6 and
             second-religion-p >= 0.2 and
             pos-of-fed = 0.2
      then   pos-of-fed = 0.4

38]   if     type-of-division = ethnic and
             first-ethnic-p >= 0.6 and
             second-ethnic-p >= 0.2 and
             pos-of-fed = 0.5
      then   pos-of-fed = 0.7

39]   if     type-of-division = ethnic and
             first-ethnic-p >= 0.6 and
             second-ethnic-p >= 0.2 and
             pos-of-fed = 0.35
      then   pos-of-fed = 0.55

40]   if     type-of-division = ethnic and
             first-ethnic-p >= 0.6 and
             second-ethnic-p >= 0.2 and
             pos-of-fed = 0.2
      then   pos-of-fed = 0.4

41]   if     type-of-division = linguistic and
             first-linguistic-p >= 0.6 and
             second-linguistic-p >= 0.2 and
             pos-of-fed = 0.5
      then   pos-of-fed = 0.7
```

```
42]   if     type-of-division = linguistic and
             first-linguistic-p >= 0.6 and
             second-linguistic-p >= 0.2 and
             pos-of-fed = 0.35
      then   pos-of-fed = 0.55

43]   if     type-of-division = linguistic and
             first-linguistic-p >= 0.6 and
             second-linguistic-p >= 0.2 and
             pos-of-fed = 0.2
      then   pos-of-fed = 0.4
```

Rules about geographical distribution

```
44]   if     geographic-grouping <> none and
             pos-of-fed < 0.2
      then   pos-of-fed =  0.25

45]   if     geographic-grouping <> none and
             pos-of-fed > 0.2
             pos-of-fed <= 0.5
      then   pos-of-fed =  0.55

46]   if     geographic-grouping <> none and
             pos-of-fed > 0.5
      then   pos-of-fed =  0.85

47]   if     geographic-grouping = racial and
             type_of_division = racial and
             pos-of-fed < 0.2
        then pos-of-fed =  0.3

48]   if     geographic-grouping = racial and
             type_of_division = racial and
             pos-of-fed > 0.2
             pos-of-fed <= 0.5
      then   pos-of-fed =  0.6

49]   if     geographic-grouping = racial and
             type_of_division = racial and
             pos-of-fed > 0.5
      then   pos-of-fed =  0.9

50]   if     geographic-grouping = religion and
             type_of_division = religion and
             pos-of-fed < 0.2
      then   pos-of-fed =  0.3
```

```
51]   if     geographic-grouping = religion and
             type_of_division = religion and
             pos-of-fed > 0.2
             pos-of-fed <= 0.5
      then   pos-of-fed =  0.6

52]   if     geographic-grouping = religion and
             type_of_division = religion and
             pos-of-fed > 0.5
      then   pos-of-fed =  0.9

53]   if     geographic-grouping = ethnic and
             type_of_division = ethnic and
             pos-of-fed < 0.2
      then   pos-of-fed =  0.3

54]   if     geographic-grouping = ethnic and
             type_of_division = ethnic and
             pos-of-fed > 0.2
             pos-of-fed <= 0.5
      then   pos-of-fed =  0.6

55]   if     geographic-grouping = ethnic and
             type_of_division = ethnic and
             pos-of-fed > 0.5
      then   pos-of-fed =  0.9

56]   if     geographic-grouping = linguistic and
             type_of_division = linguistic and
             pos-of-fed < 0.2
      then   pos-of-fed =  0.3

57]   if     geographic-grouping = linguistic and
             type_of_division = linguistic and
             pos-of-fed > 0.2
             pos-of-fed <= 0.5
      then   pos-of-fed =  0.6

58]   if     geographic-grouping = linguistic and
             type_of_division = linguistic and
             pos-of-fed > 0.5
      then   pos-of-fed =  0.9
```

Rules assessing kind of federalism or none at all

```
59]  if    type_of_division = null and
           possibility_of_federalism < 0.3
     then  federalism = FEDERALISM NOT VIABLE

60]  if    type_of_division = null and
           possibility_of_federalism > 0.3
     then  federalism = FEDERALISM MAYBE VIABLE BUT NO SIGNIFICANT
DIVISIONS IN SOCIETY TO SUSTAIN A DIVIDED STATE

61]  if    type_of_division <> null and
           possibility_of_federalism < 0.3
     then  federalism = Federalism MAYBE VIABLE but country is too small
to bear costs of federalism

62]  if    type_of_division = racial and
           possibility_of_federalism > 0.3 and
           possibility_of_federalism < 0.6
     then  federalism = Federalism POSSIBLY VIABLE along RACIAL
seperations

63]  if    type_of_division = religion and
           possibility_of_federalism > 0.3 and
           possibility_of_federalism < 0.6
     then  federalism = Federalism POSSIBLY VIABLE along RELIGION
seperations

64]  if    type_of_division = ethnic and
           possibility_of_federalism > 0.3 and
           possibility_of_federalism < 0.6
     then  federalism = Federalism POSSIBLY VIABLE along ETHNIC
seperations

65]  if    type_of_division = linguistic and
           possibility_of_federalism > 0.3 and
           possibility_of_federalism < 0.6
     then  federalism = Federalism POSSIBLY VIABLE along LINGUISTIC
seperations

66]  if    type_of_division = racial and
           possibility_of_federalism > 0.6
     then  federalism = Federalism BEST OPTION along RACIAL
seperations

67]  if    type_of_division = religion and
           possibility_of_federalism > 0.6
```

```
        then    federalism = Federalism BEST OPTION  along RELIGION
seperations

68]     if      type_of_division = ethnic and
                possibility_of_federalism > 0.6
        then    federalism = Federalism BEST OPTION  along ETHNIC
seperations

69]     if      type_of_division = linguistic and
                possibility_of_federalism > 0.6
        then    federalism = Federalism BEST OPTION along LINGUISTIC
seperations
```

Code

```
;;; Investment Knowledge Base for Joshua
;;; for use with Rule-Based Systems Exercises
;;; 6.871 Spring 2004

(in-package :ju)

; (ask [federalism iraq response] #'print-answer-with-certainty)



(defun print-answer-with-certainty (backward-support &optional (stream *standard-
output*))
  (check-type backward-support cons "backward-support from a query")
  (let ((predication (ask-database-predication backward-support)))
    (check-type predication predication "a predication from a query")
    (terpri stream)
    (ji::truth-value-case  (predication-truth-value predication)
      (*true*
       (prin1 predication stream))
      (*false*
       (write-string "[not " stream)
       (ji::print-without-truth-value predication stream)
       (write-string "]" stream)))
    (format stream " ~d" (certainty-factor predication))))



(defgeneric possesive-suffix (predication))
(defgeneric first-prompt (predication))
(defgeneric second-prompt (predication))
(defgeneric third-prompt (predication))
(defgeneric possible-values (predication))
(defgeneric get-an-answer (predication &optional stream))
(defgeneric appropriate-ptype (predication))
(defgeneric accept-prompt (predication))
(defgeneric question-prefix (predication))
(defgeneric remaining-object-string (predication))

;;; The base mixin
(define-predicate-model question-if-unknown-model () () )

(clim:define-gesture-name :my-rule :keyboard (:r :control :shift))
(clim:define-gesture-name :my-help :keyboard (:h :control :shift))
(clim:define-gesture-name :my-why :keyboard (:w :control :shift))

(defparameter *mycin-help-string*
  "
 ctrl-?  - to show the valid answers to this question
 meta-r  - to show the current rule
 meta-y  - to see why this question is asked
 meta-h  - to see this list"
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
;;; explaining why we're asking what we're asking
;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun print-why (trigger rule &optional (stream *standard-output*))
  (format stream "~%We are trying to determine ")
  (if (predicationp trigger)
    (progn (format stream "~a " (question-prefix trigger)) (say trigger stream))
    (princ trigger stream))
  (if (null rule)
    (format stream "~%This is a top level query")
    (let* ((debug-info (ji::rule-debug-info rule))
           (sub-goals (let ((ji::*known-lvs* nil))(eval (ji::rule-debug-info-context
debug-info)))))
      (format stream "~%This is being asked for by the rule ~a in order to determine:~%"
              rule)
      (format stream "~a " (question-prefix ji::*goal*)) (say ji::*goal* stream)
      (typecase sub-goals
        (ji::and-internal
         (let ((remaining-preds (rest (predication-statement sub-goals)))
               (good-answers nil)
               (remaining-stuff nil)
               (first-remaining-object-string nil))
           (labels ((do-good-preds ()
                      (when remaining-preds
                        (let ((first (pop remaining-preds)))
                          (cond
                           ((not (predicationp first))
                            (push (copy-object-if-necessary first) good-answers)
                            (do-good-preds))
                           (t
                            (let ((found-it nil))
                              (ask first
                                   #'(lambda (just)
                                       (push (ask-database-predication just) good-
answers)

                                       (setq found-it t)
                                       (do-good-preds))
                                   :do-backward-rules nil
                                   :do-questions nil)
                              (unless found-it
                                (with-statement-destructured (who value) first
                                  (declare (ignore who))
                                  (with-unification
                                    (unify trigger first)
                                    (setq first-remaining-object-string (remaining-
object-string first))
                                    (unify value first-remaining-object-string)
                                    (setq remaining-stuff
                                          (loop for pred in remaining-preds
                                                if (predicationp pred)
                                                collect (with-statement-destructured (who
value) pred
                                                            (declare (ignore who))
                                                            (unify value (if
(joshua:unbound-logic-variable-p value)
                                                                             (remaining-
object-string pred)
                                                                             (joshua:joshua-
logic-variable-value value)))
                                                            (copy-object-if-necessary
pred))
                                                else collect (copy-object-if-necessary
pred))))))))))))
             (do-good-preds))
           (loop for pred in (nreverse good-answers)
                 for first-time = t then nil
                 if first-time
                 do (format stream "~%It has already been determined whether: ")
                 else do (format stream "~%and whether: ")
                 do (say pred stream))
           (format stream "~%It remains to determine ~a ~a ~a"
                   (question-prefix trigger) first-remaining-object-string (remaining-
stuff-suffix trigger))
           (loop for pred in remaining-stuff
                 do (format stream "~%and ~a ~a ~a" (question-prefix pred) (remaining-
object-string pred) (remaining-stuff-suffix pred)))))
        (otherwise ))
      )))

(defmethod remaining-stuff-suffix ((pred predication)) "is")
(defmethod remaining-stuff-suffix ((expression cons)) "")
(defmethod predication-value-description ((pred predication)) (remaining-object-string
pred))


;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
;;;  PROTOCOL HACKING
;;;;;;;;;;;;;;;;;;;;;;;;;

(defmethod say ((expression cons) &optional (stream *standard-output*))
  (princ expression stream))

(defmethod remaining-object-string ((expression cons)) (format nil "~a" expression))

(defmethod question-prefix ((expression cons)) "whether")

(defmethod get-an-answer ((predication question-if-unknown-model) &optional (stream
*standard-output*))
  "Print the prompt for this parameter (or make one up) and read the reply."
  (fresh-line)
  (flet ((mycin-help (stream action string-so-far)
           (declare (ignore string-so-far))
           (when (member action '(:help :my-help :my-rule :my-why))
             (fresh-line stream)
             (case action
               (:my-why
                (print-why predication ji::*running-rule* stream)
                )
               (:my-rule
                (format stream "You are running the rule ~a" ji::*running-rule*))
               (:my-help
                (format stream *mycin-help-string*)
                ))
             (fresh-line stream)
             (write-string "You are being asked to enter " stream)
             (clim:describe-presentation-type (appropriate-ptype predication) stream)
             (write-char #\. stream)
             )))
    (let ((clim:*help-gestures* (list* :my-help :my-why :my-rule clim:*help-gestures*)))
      (clim:with-accept-help ((:top-level-help #'mycin-help))
        (clim:accept (appropriate-ptype predication)
                     :stream stream
                     :prompt (accept-prompt predication))))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;  Our pseudo mycin contains 3 types of predications
;;;;  boolean valued, numeric valued, and those that take one of
;;;;  a set of values
;;;;  For each type we provide say methods
;;;;   and a bunch of subordinate methods to make dialog almost English
;;;;   and to do CLIM accepts correctly
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;; boolean values
(define-predicate-model value-is-boolean-mixin () () )
```

```
(define-predicate-method (say value-is-boolean-mixin) (&optional (stream *standard-
output*))
  (with-statement-destructured (who yesno) self
    (format stream "~A~A ~A ~A"
            who (possesive-suffix self)
            (if (joshua:joshua-logic-variable-value yesno) (first-prompt self) (second-
prompt self))
            (third-prompt self))))

(defmethod remaining-object-string ((predication value-is-boolean-mixin))
  (with-statement-destructured (who value) predication
    (declare (ignore value))
    (format nil "~A ~A ~a"
            (joshua:joshua-logic-variable-value who)
            (first-prompt predication) (third-prompt predication))))

(defmethod appropriate-ptype ((predication value-is-boolean-mixin)) '(clim:member yes
no))

(defmethod accept-prompt ((predication value-is-boolean-mixin))
  (with-statement-destructured (who value) predication
    (declare (ignore value))
    (format nil "~%Is it the case that ~a~a ~a ~a"
            who (possesive-suffix predication)
            (first-prompt predication)
            (third-prompt predication))))

(defmethod question-prefix ((predication value-is-boolean-mixin)) "whether")

(defmethod possible-values ((predication value-is-boolean-mixin)) '("yes" "no"))

(defmethod remaining-stuff-suffix ((pred value-is-boolean-mixin)) "")
(defmethod predication-value-description ((pred value-is-boolean-mixin)) "foobar")


;;;; numeric values

(define-predicate-model value-is-numeric-mixin () () )
(define-predicate-method (say value-is-numeric-mixin) (&optional (stream *standard-
output*))
  (with-statement-destructured (who number) self
    (if (joshua:unbound-logic-variable-p number)
      (format stream "is ~a~a ~a"
              who (possesive-suffix self) (first-prompt self))
      (format stream "~A~A ~A is ~A ~A"
              who (possesive-suffix self)
              (first-prompt self)
              (joshua:joshua-logic-variable-value number)
              (second-prompt self)))))

(defmethod remaining-object-string ((predication value-is-numeric-mixin))
  (with-statement-destructured (who value) predication
    (declare (ignore value))
    (format nil "~A~A ~A"
            (joshua:joshua-logic-variable-value who) (possesive-suffix predication)
            (first-prompt predication))))


(defmethod appropriate-ptype ((predication value-is-numeric-mixin)) 'number)

(defmethod accept-prompt ((predication value-is-numeric-mixin))
  (with-statement-destructured (who value) predication
    (declare (ignore value))
    (format nil "~%What is ~a~a ~a"
            who (possesive-suffix predication) (first-prompt predication))))

(defmethod question-prefix ((predication value-is-numeric-mixin)) "what")


;;; variety of possible values
```

```
(define-predicate-model value-is-option-mixin ()  () )

(define-predicate-method (say value-is-option-mixin) (&optional (stream *standard-
output*))
  (with-statement-destructured (who option) self
    (format stream "~A~A ~A ~A ~A"
            who (possesive-suffix self)
            (first-prompt self)
            (second-prompt self)
            (joshua:joshua-logic-variable-value option))))

(defmethod remaining-object-string ((predication value-is-option-mixin))
  (with-statement-destructured (who value) predication
    (declare (ignore value))
    (format nil "~A~A ~A"
            (joshua:joshua-logic-variable-value who) (possesive-suffix predication)
            (first-prompt predication))))

(defmethod appropriate-ptype ((predication value-is-option-mixin)) `(member ,@(possible-
values predication)))

(defmethod accept-prompt ((predication value-is-option-mixin))
  (with-statement-destructured (who value) predication
    (declare (ignore value))
    (format nil "~%What is ~a~a ~a"
            who (possesive-suffix predication) (first-prompt predication))))

(defmethod question-prefix ((predication value-is-option-mixin)) "whether")


;;; Predicate defining macro

(defmacro define-predicate-with-ancillary-info ((pred-name mixin)
                                                &key
                                                possesive-suffix
                                                prompt1 prompt2 prompt3
                                                possible-values
                                                missing-value-prompt
                                                )
  `(eval-when (:compile-toplevel :execute :load-toplevel)
     (define-predicate ,pred-name (who value) (,mixin question-if-unknown-model cf-mixin
ltms:ltms-predicate-model))
     (defmethod possesive-suffix ((predication ,pred-name)) () ,possesive-suffix)
     (defmethod first-prompt ((predication ,pred-name)) () ',prompt1)
     (defmethod second-prompt ((predication ,pred-name)) () ',prompt2)
     ,(when prompt3 `(defmethod third-prompt ((predication ,pred-name)) () ',prompt3))
     ,(when possible-values `(defmethod possible-values ((predication ,pred-name))
',possible-values))
     ,(when missing-value-prompt `(defmethod missing-value-prompt ((predication ,pred-
name)) ',missing-value-prompt))
  ))

;;; predicates that take numeric values
(define-predicate-with-ancillary-info (population value-is-numeric-mixin)
  :possesive-suffix "'s" :prompt1 "population")
(define-predicate-with-ancillary-info (land-area value-is-numeric-mixin)
  :possesive-suffix "'s" :prompt1 "land area" :prompt2 "sq km")
(define-predicate-with-ancillary-info (pos-of-fed value-is-numeric-mixin)
  :possesive-suffix "'s" :prompt1 "possibility of federalism" :prompt2 "on a scale of 0
to 1")
(define-predicate-with-ancillary-info (racial-count value-is-numeric-mixin)
  :possesive-suffix "'s" :prompt1 "number of races" :prompt2 "is")
(define-predicate-with-ancillary-info (religion-count value-is-numeric-mixin)
  :possesive-suffix "'s" :prompt1 "number of religions practiced":prompt2 "is")
(define-predicate-with-ancillary-info (ethnic-count value-is-numeric-mixin)
    :possesive-suffix "'s" :prompt1 "number of living ethnicities":prompt2 "is")
(define-predicate-with-ancillary-info (linguistic-count value-is-numeric-mixin)
  :possesive-suffix "'s" :prompt1 "number of spoken languages" :prompt2 "is")
(define-predicate-with-ancillary-info (first-racial-p value-is-numeric-mixin)
    :possesive-suffix "'s" :prompt1 "most prevalent race's population percentage (enter
0.3 for 30%)":prompt2 "is")
(define-predicate-with-ancillary-info (second-racial-p value-is-numeric-mixin)
    :possesive-suffix "'s" :prompt1 "second most prevalent race's population percentage
(enter 0.3 for 30%)":prompt2 "is")
(define-predicate-with-ancillary-info (third-racial-p value-is-numeric-mixin)
    :possesive-suffix "'s" :prompt1 "third most prevalent race's population percentage
(enter 0.3 for 30%)":prompt2 "is")
(define-predicate-with-ancillary-info (first-religion-p value-is-numeric-mixin)
    :possesive-suffix "'s" :prompt1 "most followed religion's population percentage
(enter 0.3 for 30%)":prompt2 "is")
(define-predicate-with-ancillary-info (second-religion-p value-is-numeric-mixin)
    :possesive-suffix "'s" :prompt1 "second most followed religion's population
percentage (enter 0.3 for 30%)":prompt2 "is")
(define-predicate-with-ancillary-info (third-religion-p value-is-numeric-mixin)
    :possesive-suffix "'s" :prompt1 "third most followed religion's population percentage
(enter 0.3 for 30%)":prompt2 "is")
(define-predicate-with-ancillary-info (first-ethnic-p value-is-numeric-mixin)
    :possesive-suffix "'s" :prompt1 "most populous ethnicity's population percentage
(enter 0.3 for 30%)":prompt2 "is")
(define-predicate-with-ancillary-info (second-ethnic-p value-is-numeric-mixin)
    :possesive-suffix "'s" :prompt1 "second most populous ethnicity's population
percentage (enter 0.3 for 30%)":prompt2 "is")
(define-predicate-with-ancillary-info (third-ethnic-p value-is-numeric-mixin)
    :possesive-suffix "'s" :prompt1 "third most populous ethnicity's population
percentage (enter 0.3 for 30%)":prompt2 "is")
(define-predicate-with-ancillary-info (first-linguistic-p value-is-numeric-mixin)
    :possesive-suffix "'s" :prompt1 "most spoken language's population percentage (enter
0.3 for 30%)":prompt2 "is")
(define-predicate-with-ancillary-info (second-linguistic-p value-is-numeric-mixin)
    :possesive-suffix "'s" :prompt1 "second most spoken language's population percentage
(enter 0.3 for 30%)":prompt2 "is")
(define-predicate-with-ancillary-info (third-linguistic-p value-is-numeric-mixin)
    :possesive-suffix "'s" :prompt1 "third most spoken language's population percentage
(enter 0.3 for 30%)":prompt2 "is")
(define-predicate-with-ancillary-info (first-p value-is-numeric-mixin)
    :possesive-suffix "'s" :prompt1 "most important diversity's highest population
percentage (enter 0.3 for 30%)":prompt2 "is")
(define-predicate-with-ancillary-info (second-p value-is-numeric-mixin)
    :possesive-suffix "'s" :prompt1 "most important diversity's second highest population
percentage (enter 0.3 for 30%)":prompt2 "is")
(define-predicate-with-ancillary-info (third-p value-is-numeric-mixin)
    :possesive-suffix "'s" :prompt1 "most important diversity's third highest population
percentage (enter 0.3 for 30%)":prompt2 "is")

;;; Predicates that take one of a set of values
(define-predicate-with-ancillary-info (type-of-division value-is-option-mixin)
  :possesive-suffix "'s" :prompt1 "most prominent type of division
(none/racial/religion/ethnic/linguistic)" :prompt2 "is"
  :possible-values (none racial religion ethnic linguistic))
(define-predicate-with-ancillary-info (geographic-grouping value-is-option-mixin)
  :possesive-suffix "'s" :prompt1 "geographical grouping along lines of community
(none/racial/religion/ethnic/linguistic)" :prompt2 "is"
  :possible-values (none racial religion ethnic linguistic))
(define-predicate-with-ancillary-info (federalism value-is-option-mixin)
  :possesive-suffix "'s" :prompt1 "option on federalism" :prompt2 "is"
  :possible-values (not-viable maybe-viable-in-undivided-society maybe-viable-but-
country-very-small possibly-viable-racially possibly-viable-along-religion possibly-
viable-ethnically possibly-viable-along-language best-option-along-race best-option-
along-religion best-option-along-ethnicity best-option-along-language))


;; using this model, the system will ask the user any time
;; it needs a specific fact to continue backward chaining.

;;; we should only be asking a question under the following
;;; circumstances:
;;;
;;; the predication being asked contains no logic variables
;;; eg. [has-health-insurance matt yes], not
;;; [has-health-insurance matt ?x]
;;;
;;; AND
;;;
;;; that predication is not already in the database
```

```
;;;
;;; AND
;;;
;;; any other predication matching the predicate and ?who
;;; eg. [has-health-insurance matt no] is not already in the
;;; database.
;;;
;;; AND
;;;
;;; there is no rule we can use to find out the answer
;;;
;;; this can be told by check [known [has-health-insurance matt ?]]


(define-predicate already-known (predicate object))

;;; if after doing the normal processing nothing is found
;;; then finally ask the guy a question if appropriate
(define-predicate-method (ask question-if-unknown-model) (intended-truth-value
continuation do-backward-rules do-questions)
  (let ((answers nil)
        (predicate (predication-predicate self)))
    (flet ((my-continuation (bs)
             (let* ((answer (ask-query bs))
                    (database-answer (insert (copy-object-if-necessary answer))))
               (pushnew database-answer answers))))
      (with-statement-destructured (who value) self
        (declare (ignore value))
        (with-unbound-logic-variables (value)
          (let ((predication `[,predicate ,who ,value]))
            ;; first see if there's an answer already in the database
            ;; may want to change this to asserting already-know predication, but I'm
trying to avoid that
            (ask-data predication intended-truth-value #'my-continuation)
            (unless answers
              ;; Now go get stuff from rules.
              (when do-backward-rules
                (ask-rules predication intended-truth-value #'my-continuation do-
questions))
              ;; now go hack questions
              (unless answers
                (when do-questions
                  (ask-questions predication intended-truth-value #'my-continuation))))))
        ;; if he's doing a raws database fetch, don't ask
        (when (and (null answers) (or do-backward-rules do-questions))
          (unless (joshua:unbound-logic-variable-p who)
            (let* ((answer (get-an-answer self))
                   (database-answer (tell `[,predicate ,who ,answer]
                                          :justification '((user-input 1.0)))))
              (pushnew database-answer answers))))))
    (loop for answer in answers
          when (eql (predication-truth-value answer) intended-truth-value)
          do (with-stack-list (just self intended-truth-value answer)
               (with-unification
                 (unify self answer)
                 (funcall continuation just)))))
  ;; make it clear that there is no interesting return value
  (values))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                             ;;;
;;; Inference Rules (For importance, higher values go first.)   ;;;
;;;                                                         ;;;
;;;                                                             ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;RULES ABOUT: population and land-area minimum requirements

(defrule population-low-cutoff (:backward :certainty 0.5 :importance 90)
  if [and [population ?who ?x]
  [land-area ?who ?y]
  (and(< ?x 10000000)(< ?y 100000))]
  then [pos-of-fed ?who 0.2])


(defrule population-medium-cutoff (:backward :certainty 0.5 :importance 89)
  if [and [population ?who ?x]
  [land-area ?who ?y]
  (and (< ?x 10000000)(>= ?y 100000))]
  then [pos-of-fed ?who 0.35])


(defrule population-medium-cutoff2 (:backward :certainty 0.5 :importance 88)
  if [and [population ?who ?x]
  [land-area ?who ?y]
  (and (>= ?x 10000000)(< ?y 100000))]
  then [pos-of-fed ?who 0.35])


(defrule population-high-cutoff (:backward :certainty 0.5 :importance 87)
  if [and [population ?who ?x]
  [land-area ?who ?y]
  (and(>= ?x 10000000)(>= ?y 100000))]
  then [pos-of-fed ?who 0.5])

;;;;RULES ABOUT: Diversity typecasting


(defrule race-diversity (:backward :certainty 0.8 :importance 85)
  if [and [racial-count ?who ?x]
  [first-racial-p ?who ?y]
  (and (> ?x 2)(< ?y 0.9))]
  then [type-of-division ?who racial])


(defrule none-check-1 (:backward :certainty 0.00001 :importance 84)
  if [and [racial-count ?who ?x]
          (<= ?x 2)]
       then [type-of-division ?who none])

(defrule religion-none (:backward :certainty 0.8 :importance 83)
  if [and [religion-count ?who ?x]
       [first-religion-p ?who ?y]
       (and(> ?x 2)(< ?y 0.9))]
       then [type-of-division ?who religion])



(defrule religion-racial (:backward :certainty 0.9 :importance 82)
  if [and [religion-count ?who ?x]
       [first-religion-p ?who ?y]
       [type-of-division ?who racial]
       [first-racial-p ?who ?w]
  (and(> ?x 2)(< ?y 0.9)(< ?y ?w))]
  then [type-of-division ?who religion])

(defrule ethnic-none (:backward :certainty 0.8 :importance 81)
  if [and
        [ethnic-count ?who ?x]
       [first-ethnic-p ?who ?y]
       (and(> ?x 2)
            (< ?y 0.9))]
  then [type-of-division ?who ethnic])

(defrule ethnic-racial (:backward :certainty 0.9 :importance 80)
  if [and [ethnic-count ?who ?x]
       [first-ethnic-p ?who ?y]
       [type-of-division ?who racial]
       [first-racial-p ?who ?w]
 (and(> ?x 2)
  (< ?y 0.9)
  (< ?y ?w))]
  then [type-of-division ?who ethnic])
```

```
(defrule ethnic-religion (:backward :certainty 0.9 :importance 79)
  if [and [ethnic-count ?who ?x]
       [first-ethnic-p ?who ?y]
       [type-of-division ?who religion]
       [first-religion-p ?who ?w]
 (and (> ?x 2)
  (< ?y 0.9)
  (< ?y ?w))]
  then [type-of-division ?who ethnic])

(defrule linguistic-none (:backward :certainty 0.8 :importance 78)
  if [and [linguistic-count ?who ?x]
       [first-linguistic-p ?who ?y]
  (and(> ?x 2)
  (< ?y 0.9))]
  then [type-of-division ?who linguistic])

(defrule linguistic-racial (:backward :certainty 0.9 :importance 77)
  if [and [linguistic-count ?who ?x]
       [first-linguistic-p ?who ?y]
       [type-of-division ?who racial]
       [first-racial-p ?who ?w]
  (and(> ?x 2)
  (< ?y 0.9)
  (< ?y ?w))]
  then [type-of-division ?who linguistic])

(defrule linguistic-religion (:backward :certainty 0.9 :importance 76)
  if [and [linguistic-count ?who ?x]
       [first-linguistic-p ?who ?y]
       [type-of-division ?who religion]
       [first-religion-p ?who ?w]
 (and (> ?x 2)
  (< ?y 0.9)
  (< ?y ?w))]
  then [type-of-division ?who linguistic])

(defrule linguistic-ethnic (:backward :certainty 0.9 :importance 75)
  if [and [linguistic-count ?who ?x]
       [first-linguistic-p ?who ?y]
       [type-of-division ?who ethnic]
       [first-ethnic-p ?who ?w]
  (and(> ?x 2)
      (< ?y 0.9)

  (< ?y ?w))]
  then [type-of-division ?who linguistic])

(defrule none-final (:backward :certainty 0.7 :importance 74)
 if [or  [and  [linguistic-count ?who ?w]
                [religion-count ?who ?y]
                [racial-count ?who ?x]
                [ethnic-count ?who ?z]
               (and (< ?x 2)
                    (< ?y 2)
                    (< ?z 2)
                    (< ?w 2))]
          [and [first-linguistic-p ?who ?d]
         [first-racial-p ?who ?a]
         [first-religion-p ?who ?b]
         [first-ethnic-p ?who ?c]
         (and (> ?a .9) (> ?b .9)(> ?c .9)(> ?d .9))]]
  then [type-of-division ?who none])



;;;; Rules about percentages

(defrule none-p-1 (:backward :certainty 1.0 :importance 73)
  if [type-of-division ?who none]
  then [first-p ?who 1])
```

```
(defrule none-p-2 (:backward :certainty 1.0 :importance 73)
  if [type-of-division ?who none]
  then  [second-p ?who 0])

(defrule none-p-3 (:backward :certainty 1.0 :importance 73)
  if [type-of-division ?who none]
  then [third-p ?who 0]])

(defrule racial-p-h (:backward :certainty 0.7 :importance 73)
  if [and[type-of-division ?who racial]
        [first-racial-p ?who ?x]
       [second-racial-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (<= ?x 0.5)
            (>= ?y 0.3)
            (= ?z 0.5))]
  then [pos-of-fed ?who 0.8])

(defrule racial-p-m (:backward :certainty 0.7 :importance 73)
  if [and[type-of-division ?who racial]
        [first-racial-p ?who ?x]
       [second-racial-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (<= ?x 0.5)
            (>= ?y 0.3)
            (= ?z 0.35))]
  then [pos-of-fed ?who 0.65])


(defrule racial-p-l (:backward :certainty 0.7 :importance 73)
  if [and[type-of-division ?who racial]
        [first-racial-p ?who ?x]
       [second-racial-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (<= ?x 0.5)
            (>= ?y 0.3)
            (= ?z 0.2))]
  then [pos-of-fed ?who 0.5])


(defrule religion-p-h (:backward :certainty 0.7 :importance 72)
  if [and[type-of-division ?who religion]
        [first-religion-p ?who ?x]
       [second-religion-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (<= ?x 0.5)
            (>= ?y 0.3)
            (= ?z 0.5))]
  then [pos-of-fed ?who 0.8])


(defrule religion-p-m (:backward :certainty 0.7 :importance 72)
  if [and[type-of-division ?who religion]
        [first-religion-p ?who ?x]
       [second-religion-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (<= ?x 0.5)
            (>= ?y 0.3)
            (= ?z 0.35))]
            then [pos-of-fed ?who 0.65])

(defrule religion-p-l (:backward :certainty 0.7 :importance 72)
  if [and[type-of-division ?who religion]
        [first-religion-p ?who ?x]
       [second-religion-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (<= ?x 0.5)
            (>= ?y 0.3)
            (= ?z 0.2))]
  then [pos-of-fed ?who 0.5])
```

```
(defrule ethnic-p-h (:backward :certainty 0.7 :importance 71)
  if [and[type-of-division ?who ethnic]
        [first-ethnic-p ?who ?x]
       [second-ethnic-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (<= ?x 0.5)
            (>= ?y 0.3)
            (= ?z 0.5))]
  then [pos-of-fed ?who 0.8])

(defrule ethnic-p-m (:backward :certainty 0.7 :importance 71)
  if [and[type-of-division ?who ethnic]
        [first-ethnic-p ?who ?x]
       [second-ethnic-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (<= ?x 0.5)
            (>= ?y 0.3)
            (= ?z 0.35))]
            then [pos-of-fed ?who 0.65])

(defrule ethnic-p-l (:backward :certainty 0.7 :importance 71)
  if [and[type-of-division ?who ethnic]
        [first-ethnic-p ?who ?x]
       [second-ethnic-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (<= ?x 0.5)
            (>= ?y 0.3)
            (= ?z 0.2))]
  then [pos-of-fed ?who 0.5])

(defrule linguistic-p-h (:backward :certainty 0.7 :importance 70)
  if [and[type-of-division ?who linguistic]
      [first-linguistic-p ?who ?x]
       [second-linguistic-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (<= ?x 0.5)
            (>= ?y 0.3)
            (= ?z 0.5))]
            then [pos-of-fed ?who 0.8])

(defrule linguistic-p-m (:backward :certainty 0.7 :importance 70)
  if [and[type-of-division ?who linguistic]
      [first-linguistic-p ?who ?x]
       [second-linguistic-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (<= ?x 0.5)
            (>= ?y 0.3)
            (= ?z 0.35))]
            then [pos-of-fed ?who 0.65])

(defrule linguistic-p-l (:backward :certainty 0.7 :importance 70)
  if [and[type-of-division ?who linguistic]
      [first-linguistic-p ?who ?x]
       [second-linguistic-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (<= ?x 0.5)
            (>= ?y 0.3)
            (= ?z 0.2))]
  then [pos-of-fed ?who 0.5])

(defrule racial-p-2-h (:backward :certainty 0.7 :importance 73)
  if [and[type-of-division ?who racial]
        [first-racial-p ?who ?x]
       [second-racial-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (>= ?x 0.6)
            (>= ?y 0.2)
            (= ?z 0.5))]
  then [pos-of-fed ?who 0.7])
```

```
(defrule racial-p-2-m (:backward :certainty 0.7 :importance 73)
  if [and[type-of-division ?who racial]
        [first-racial-p ?who ?x]
       [second-racial-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (>= ?x 0.6)
            (>= ?y 0.2)
            (= ?z 0.35))]
            then [pos-of-fed ?who 0.55])


(defrule racial-p-2-l (:backward :certainty 0.7 :importance 73)
  if [and[type-of-division ?who racial]
        [first-racial-p ?who ?x]
       [second-racial-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (>= ?x 0.6)
            (>= ?y 0.2)
            (= ?z 0.2))]
            then [pos-of-fed ?who 0.4])


(defrule religion-p-2-h (:backward :certainty 0.7 :importance 72)
  if [and[type-of-division ?who religion]
        [first-religion-p ?who ?x]
       [second-religion-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (>= ?x 0.6)
            (>= ?y 0.2)
            (= ?z 0.5))]
   then [pos-of-fed ?who 0.7])


(defrule religion-p-2-m (:backward :certainty 0.7 :importance 72)
  if [and[type-of-division ?who religion]
        [first-religion-p ?who ?x]
       [second-religion-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (>= ?x 0.6)
            (>= ?y 0.2)
            (= ?z 0.35))]
            then [pos-of-fed ?who 0.55])

(defrule religion-p-2-l (:backward :certainty 0.7 :importance 72)
  if [and[type-of-division ?who religion]
        [first-religion-p ?who ?x]
       [second-religion-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (>= ?x 0.6)
            (>= ?y 0.2)
            (= ?z 0.2))]
            then [pos-of-fed ?who 0.4])

(defrule ethnic-p-2-h (:backward :certainty 0.7 :importance 71)
  if [and[type-of-division ?who ethnic]
        [first-ethnic-p ?who ?x]
       [second-ethnic-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (>= ?x 0.6)
            (>= ?y 0.2)
            (= ?z 0.5))]
            then [pos-of-fed ?who 0.7])

(defrule ethnic-p-2-m (:backward :certainty 0.7 :importance 71)
  if [and[type-of-division ?who ethnic]
        [first-ethnic-p ?who ?x]
       [second-ethnic-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (>= ?x 0.6)
            (>= ?y 0.2)
            (= ?z 0.35))]
            then [pos-of-fed ?who 0.55])
```

```
(defrule ethnic-p-2-l (:backward :certainty 0.7 :importance 71)
  if [and[type-of-division ?who ethnic]
        [first-ethnic-p ?who ?x]
       [second-ethnic-p ?who ?y]
       [pos-of-fed ?who ?z]
       (and (>= ?x 0.6)
            (>= ?y 0.2)
            (= ?z 0.2))]
            then [pos-of-fed ?who 0.4])


(defrule linguistic-p-2-h (:backward :certainty 0.7 :importance 70)
  if [and[type-of-division ?who linguistic]
      [first-linguistic-p ?who ?x]
      [second-linguistic-p ?who ?y]
      [pos-of-fed ?who ?z]
       (and (>= ?x 0.6)
            (>= ?y 0.2)
            (= ?z 0.5))]
  then [pos-of-fed ?who 0.7])

(defrule linguistic-p-2-m (:backward :certainty 0.7 :importance 70)
  if [and[type-of-division ?who linguistic]
      [first-linguistic-p ?who ?x]
      [second-linguistic-p ?who ?y]
      [pos-of-fed ?who ?z]
       (and (>= ?x 0.6)
            (>= ?y 0.2)
            (= ?z 0.35))]
            then [pos-of-fed ?who 0.55])

(defrule linguistic-p-2-l (:backward :certainty 0.7 :importance 70)
  if [and[type-of-division ?who linguistic]
      [first-linguistic-p ?who ?x]
      [second-linguistic-p ?who ?y]
      [pos-of-fed ?who ?z]
       (and (>= ?x 0.6)
            (>= ?y 0.2)
            (= ?z 0.2))]
  then [pos-of-fed ?who 0.4])



;;;;; Rules about geographic grouping

(defrule geo-grouping-check-l (:backward :certainty 0.8 :importance 67)
  if [and (not [geographic-grouping ?who none])
     [pos-of-fed ?who ?z]
     (<= ?z 0.2)]
  then  [pos-of-fed ?who 0.25])

(defrule geo-grouping-check-m (:backward :certainty 0.8 :importance 66)
  if [and (not [geographic-grouping ?who none])
     [pos-of-fed ?who ?z]
     (and (<= ?z 0.5)(> ?z 0.2))]
     then      [pos-of-fed ?who 0.55])

(defrule geo-grouping-check-h (:backward :certainty 0.8 :importance 66)
  if [and (not [geographic-grouping ?who none])
     [pos-of-fed ?who ?z]
     (> ?z 0.5)]
  then  [pos-of-fed ?who 0.85])

(defrule racial-grouping-l (:backward :certainty 0.9 :importance 66)
  if [and [geographic-grouping ?who racial]
       [type-of-division ?who racial]
       [pos-of-fed ?who ?z]
       (<= ?z 0.2)]]
  then  [pos-of-fed ?who 0.3])

(defrule racial-grouping-m (:backward :certainty 0.9 :importance 66)
  if [and [geographic-grouping ?who racial]
        [type-of-division ?who racial]
        [pos-of-fed ?who ?z]
          (and (<= ?z 0.5)(> ?z 0.2))]
          then  [pos-of-fed ?who 0.6]])

(defrule racial-grouping-h (:backward :certainty 0.9 :importance 66)
  if [and [geographic-grouping ?who racial]
        [type-of-division ?who racial]
        [pos-of-fed ?who ?z]
        (> ?z 0.5)]
  then  [pos-of-fed ?who 0.9]])

(defrule religion-grouping-l (:backward :certainty 0.9 :importance 65)
  if [and [geographic-grouping ?who religion]
         [type-of-division ?who religion]
        [pos-of-fed ?who ?z]
        (<= ?z 0.2)]
  then  [pos-of-fed ?who 0.3]])

(defrule religion-grouping-m (:backward :certainty 0.9 :importance 65)
  if [and [geographic-grouping ?who religion]
         [type-of-division ?who religion]
        [pos-of-fed ?who ?z]
         (and (<= ?z 0.5)(> ?z 0.2))]
then    [pos-of-fed ?who 0.6]])


(defrule religion-grouping-h (:backward :certainty 0.9 :importance 65)
  if [and [geographic-grouping ?who religion]
         [type-of-division ?who religion]
        [pos-of-fed ?who ?z]
        (> ?z 0.5)]
  then  [pos-of-fed ?who 0.9]])



(defrule ethnic-grouping-l (:backward :certainty 0.9 :importance 64)
  if [and [geographic-grouping ?who ethnic]
        [type-of-division ?who ethnic]
        [pos-of-fed ?who ?z]
        (<= ?z 0.2)]
  then  [pos-of-fed ?who 0.3]])

(defrule ethnic-grouping-m (:backward :certainty 0.9 :importance 64)
  if [and [geographic-grouping ?who ethnic]
          [type-of-division ?who ethnic]
          [pos-of-fed ?who ?z]
          (and
           (<= ?z 0.5)
           (> ?z 0.2))]
  then  [pos-of-fed ?who 0.6]])



(defrule ethnic-grouping-h (:backward :certainty 0.9 :importance 64)
  if [and [geographic-grouping ?who ethnic]
        [type-of-division ?who ethnic]
        [pos-of-fed ?who ?z]
          (> ?z 0.5)]
  then  [pos-of-fed ?who 0.9]])



(defrule linguistic-grouping-l (:backward :certainty 0.9 :importance 63)
  if [and [geographic-grouping ?who linguistic]
      [type-of-division ?who linguistic]
      [pos-of-fed ?who ?z]
      (<= ?z 0.2)]
  then  [pos-of-fed ?who 0.3]])

(defrule linguistic-grouping-m (:backward :certainty 0.9 :importance 63)
  if [and [geographic-grouping ?who linguistic]
     [type-of-division ?who linguistic]
     [pos-of-fed ?who ?z]
     (and (<= ?z 0.5)(> ?z 0.2))]
  then  [pos-of-fed ?who 0.6])



(defrule linguistic-grouping-h (:backward :certainty 0.9 :importance 63)
  if [and [geographic-grouping ?who linguistic]
     [type-of-division ?who linguistic]
     [pos-of-fed ?who ?z]
     (> ?z 0.5)]
     then [pos-of-fed ?who 0.9])




;;;; Rules about final decision based on pre-calculated factors


(defrule not-viable (:backward :certainty 1.0 :importance 62)
  if [and [type-of-division ?who none]
       [pos-of-fed ?who ?y]
       (< ?y 0.3)]
  then [federalism ?who not-viable])


(defrule maybe-viable-1 (:backward :certainty 1.0 :importance 61)
  if [and [type-of-division ?who none]
       [pos-of-fed ?who ?y]
       (and (> ?y 0.3)(< ?y 0.6))]
  then [federalism ?who maybe-viable-in-undivided-society])

(defrule maybe-viable-2 (:backward :certainty 1.0 :importance 59)
  if [and (not[type-of-division ?who none])
       [pos-of-fed ?who ?y]
       (> ?y 0.3)]
  then [federalism ?who maybe-viable-but-country-very-small])


(defrule possibly-viable-racially (:backward :certainty 0.9 :importance 58)
  if [and [type-of-division ?who racial]
       [pos-of-fed ?who ?y]
       (and (> ?y 0.3)(< ?y 0.6))]
  then [federalism ?who possibly-viable-racially])

(defrule possibly-viable-along-religion (:backward :certainty 0.9 :importance 57)
  if [and [type-of-division ?who religion]
       [pos-of-fed ?who ?y]
       (and (> ?y 0.3)(< ?y 0.6))]
  then [federalism ?who possibly-viable-along-religion])

(defrule possibly-viable-ethnically (:backward :certainty 0.9 :importance 56)
  if [and [type-of-division ?who ethnic]
       [pos-of-fed ?who ?y]
       (and (> ?y 0.3)(< ?y 0.6))]
  then [federalism ?who possibly-viable-ethnically])

(defrule possibly-viable-along-language (:backward :certainty 0.9 :importance 55)
  if [and [type-of-division ?who linguistic]
       [pos-of-fed ?who ?y]
       (and (> ?y 0.3)(< ?y 0.6))]
  then [federalism ?who possibly-viable-along-language])

(defrule best-option-along-race (:backward :certainty 1.0 :importance 54)
  if [and [type-of-division ?who racial]
       [pos-of-fed ?who ?y]
       (> ?y 0.6)]
  then [federalism ?who best-option-along-race])

(defrule best-option-along-religion (:backward :certainty 1.0 :importance 53)
  if [and [type-of-division ?who religion]
       [pos-of-fed ?who ?y]
       (> ?y 0.6)]
  then [federalism ?who best-option-along-religion])

(defrule best-option-along-ethnicity (:backward :certainty 1.0 :importance 52)
  if [and [type-of-division ?who ethnic]
       [pos-of-fed ?who ?y]
       (> ?y 0.6)]
  then [federalism ?who best-option-along-ethnicity])

(defrule best-option-along-language (:backward :certainty 1.0 :importance 51)
  if [and [type-of-division ?who linguistic]
       [pos-of-fed ?who ?y]
       (> ?y 0.6)]
  then [federalism ?who best-option-along-language])

;;;;;;;;;;;;;;;;-------------------------



(defun rules-concluding-predicate (pred)
  (let ((answers nil))
    (map-over-backward-rule-triggers `[,pred ? ?]
                                    #'(lambda (trigger) (pushnew (ji::backward-trigger-
rule trigger) answers)))
    answers))


(defun predicates-rule-relies-on (rule)
  (let ((answers nil))
    (labels ((do-one-level (stuff)
                 (let ((connective (when (predication-maker-p stuff) (predication-maker-
predicate stuff))))
                   (case connective
                     ((and or)
                      (with-predication-maker-destructured (&rest more-stuff) stuff
                        (loop for thing in more-stuff
                              do (do-one-level thing))))
                     ((nil))
                     (otherwise
                      (pushnew connective answers))
                     ))))
        (do-one-level (ji::rule-debug-info-context (ji::rule-debug-info rule))))
    answers))


(defun graph-rule-tree (predicates &key (orientation :vertical) (size :small) (stream
*standard-output*))
  (terpri stream)
  (clim:with-text-size (stream size)
    (clim:format-graph-from-roots
     (loop for pred in predicates
           collect (list 'predicate pred))
     #'(lambda (thing stream)
         (destructuring-bind (type name) thing
           (case type
             (predicate
              (clim:surrounding-output-with-border (stream)
                (princ name stream)))
             (rule
              (clim:surrounding-output-with-border (stream :shape :oval)
                (princ name stream))))))
     #'(lambda (thing)
         (destructuring-bind (type name) thing
           (case type
             (predicate (loop for r in (rules-concluding-predicate name)
                              collect (list 'rule r)))
             (rule (loop for p in (predicates-rule-relies-on name)
                         collect (list 'predicate p))))))
     :stream stream
     :orientation orientation
     :merge-duplicates t
     :duplicate-test #'equal)))

(clim-env::define-lisp-listener-command (com-graph-rules :name t)
        ((predicates `(clim:sequence (member ,@(loop for pred being the hash-keys of
        ji::*all-predicates* collect pred)))
       :prompt "A sequence of predicates")
       &key
        (orientation `(clim:member :vertical :horizontal) :default :vertical)
       (size `(clim:member :tiny :very-small :small :normal :large :very-large :huge)
       :default :small)
       (to-file 'clim:pathname :default nil)
       (page-orientation '(clim:member :portrait :landscape)
       :default :portrait
       :prompt "If to file, print in portrait or landscape format")
        (multi-page 'clim:boolean :default nil :prompt "If to file, segment into multiple
        pages")
       (scale-to-fit 'clim:boolean :default nil :prompt "If to file, scale to fit one
       page"))

(if to-file
        (with-open-file (file to-file :direction :output :if-exists :supersede :if-does-
        not-exist :create)
                (clim:with-output-to-postscript-stream (stream file
                                                        :multi-page multi-page
                                                        :scale-to-fit scale-to-fit
                                                        :orientation page-orientation)
        (graph-rule-tree predicates :orientation orientation :size size :stream stream)))
     (graph-rule-tree predicates :orientation orientation :size size)))
```

Bibliography

Patterns of Democracy : Government Forms and Performance in Thirty-Six Countries by Arend Lijphart

Comparative constitutional Engineering by Giovanni Sartori

Comparative Politics Today: A World View, Eighth Edition by Gabriel A. Almond, et al

The value patterns of democracy: A case study in comparative analysis (Reprint / Institute of Industrial Relations and Institute of International Studies) by Seymour Martin Lipset

Politics in Western European democracies: patterns and problems by Gary C Byrne