Chapter 15
Political Logic Programming (PLP)

15- Polit [Political] Logic Programming (PLP)

15.1- Prologue - Process logic is an analysis of the inherent logic of a process in terms of entity types, relationship, and attribute involved, the conditions constraining the execution of it sub processes, and the algorithms used. Logic analysis an programming consists of:

- Declaring some facts about polit-objects and their relationship.
- Defining some rules about polit-objects and their relationships.
- Asking questions about polit-objects and their relationship.

Logic analysis and programming is used for solving political problem: that involve polit-objects and the relationship between polit-objects.

Logic lies at the heart of computer programming. In many ways, a programming language is simply an implementation of a special form of logic.

The PROLOG language, which was chosen as one of the bases of the Japanese fifth Generation Computer Project, provides a means for knowledge representation using the logic-oriented paradigm. PROLOG was originally designed to assist in the analysis and interpretation of natural language. It caters for the expression of logical propositions including universal and existential quantifiers in the form of particular types of clauses called Horn Clauses. An existing set of declared propositions represents a set of axioms , and al automated resolution-refutation proof procedure enables the user to enter as a query some statement which is interpreted as the statement to be proved from the axioms. PROLOG adds the refutation of this statement to the axiom set, and automatically applies the resolution principle to attempt to generate a contradiction.

The CIPSE-PLP is a knowledge-based simulation system for generating and analyzing alternative scenarios in geopolitical situations, events, and processes. Given an event or process (or set of events or processes), CIPSE-PLP simulates the responses of various element (Countries, lands, leaders, radical group, etc.) in context specified by the particular scenario. These responses, in turn, generate new events which no new events can be generated or until the termination criteria of the simulation are met.

CIPSE-PLP is intended to become a tool that aids intelligence analysts by evaluating the likely consequences of critical events.

FIG.15.1. IS AN EXAMPLE FOR MATERIALS OF POLIT LOGIC PROGRAMMING.

15.2- List of Polit-Logic Programming (PLP):

- Polit-logic analysis and model
- Logic program
- Polit-logic shells
- Learning from Sentences
- Learning from predicate.

15.3- Basic Definitions in PLP:

Relation- It is easy to Prolog to define a relation such as the (conflict, parties relation), by stating the n-tuples of objects that satisfy the relation.

The user can easily query the Prolog system about relation defined in the program.

Clause- A Prolog program consists of clauses. Each clause terminates with a full stop. Prologue programs can be extended b) simply adding new clauses. Clauses built using predicates and logical connectives and are allowed to contain variables that are assumed universally quantified. Prolog clauses are three types: **facts, rules** and **questions.**

- **Facts;** declare things that are always, unconditionally true. Facts are clausee that have a head and the empty body.
- **Rules;** declare things that are true depending on a given condition. Rules have the head and the (non-empty) body.
- **Questions;** by means of questions the user can ask the program what things are true. Queering about relations, by means of questions, resembles queering a database. Prolog's answer to a question consists of a set of objects that satisfy the question. Questions only have the body.

Prolog clauses consist of the **head** and **body.** The body is a list of goals separated by commas. Commas are understood as conjunctions.

Objects and variables - .The arguments of relations can (among other things) be: concrete objects, or constants (such as tom and ana), or general objects such as X and Y. Objects of the first in our program are called atoms. Objects of the second kind are called variables.

Predicate - The discrete mathematics and AI literature is somewhat inconsistent in usage of the terms "predicate" and "predicate functions". A predicate is a parameterized proposition, that is, a proposition with variables. A predicate function on a set A is a function that maps elements of A into the set {TRUE, FALSE}.
Predicate logic is a branch of logic that allows modelling of the truth of statement based upon the values assumed by portions (0r phrases) of the statements.

Goal - Questions to the system consist of one or more goals. A sequence of goals, such as:
Country (X, paris), country (X, dehli) means the conjunction of the goals:

X is a country of Paris, and
X is a country of Dehli.

The word 'goal' is used because Prolog accepts questions as goals that are to be satisfied.
An answer to a question can be either positive or negative: depending on whether the corresponding goal is satisfied or not. In the case of a positive answer we say that the corresponding goal was satisfied and that the goal succeeded. Otherwise the goal was insatiable and it failed.

Two special goals are sometimes useful: **true** always succeeds, **fail** always fails.

Rules - Rule clauses are typified by the following example:
has (X, atombomb) : - need_control (X).

The above statement is known as a Horn clause in predicate calculus and forms the basis for PROLOG rule-based inference. '
A rule is a clause with a nonempty head and tail. PROLOG clauses comprising rules are assumed to be TRUE, therefore the ·head of a rule is T**RUE** if the tailor body may be proved **TRUE**.

Procedure - Is a set of clauses about the same relation.

Predecessor - This relation will be defined in terms of the parent relation. This relation will be defined in terms of the paren1 relation. The first rule will define the direct predecessors and the second rule the indirect predecessors.

Declarative and Procedural - Two types of meaning of programs are distinguished: declarative and procedural. The declarative view is advantageous from the programming point of view. Nevertheless, the procedural details often have to be considered by the programmer as well.

Atoms - Represented by an unbroken string of characters, and is the smallest strings that can manipulated.

Variables - Prolog uses logical variables (written in uppercase letters) to enable more powerful kinds of questioning through existential queries. Existential queries are queries containing one or more logical variables.

Data objects - Following figure shows a classification of date objects in Prolog. The Prolog system recognizes the type of at object in the program by its syntactic form. This is possible because the syntax of Prolog specifies different forms for each type of data objects.
Simple objects in Prolog are atoms, variable and numbers. Structure (objects, or structures,. are used to represent objects that have several components. The type of object is recognized entirely by its syntactic form.

Lists - Lists are a useful and popular data structure. In PROLOG, (list is one form of structure. PROLOG permits through unification the manipulation of data in list format.
Lists are permitted in PROLOG and consist of elements separate (by commas and enclosed in brackets [ ], i.e..

[a,b,c,d]

is a list of elements a thru d.
Lists may be used as arguments to predicates. Elements of a list may be other terms, including variables, constants and other lists.

Structures - Structured objects (or simply structure) are object: that have several components. The components themselves can, in turn, be structures. Structures are constructed by means on functor. Each functor is defined by its name.
Structures can be naturally pictured as tree. Prolog can be viewed as a language for processing trees.

Propositional Logic - propositional logic deals with the determination of the truthfulness or falseness of various propositition s. A proposition is a properly formed statement that is e1ther true or false. Propositional logic uses operators to connect propositions. Here are the most common operators:
AND
OR
NOT
IMPLIES
EQUIVALENT

Predicate Calculus - Predicate calculus has nothing to do with the branch of higher mathematics called calculus. Predicate calculus, sometimes called predicate logic, is simply an extension of propositional logic that allows statement about individual objects t< be examined for their truth value.
The basis for predicate calculus is the predicate, which essentially is a function that either a value of true or false depending upon its argument. A simple predicate might be polit_expert.

Database - A data base can be naturally represented as a set of prolog facts.

15.4- Symbolic Logic - The discipline of mathematical logic, in particular symbolic logic, is a subset of the discipline of discrete mathematics. Logic can be used to express statements, to validly infer additional statements, and to rigorously prove (or disprove: statements.
PLP is related to mathematical logic, so its syntax and meaning can be specified most concisely with references to logic.
A statement is a declarative sentence that is either TRUE on FALSE. A compound statement if formed by using logical connectives: such as "and", "not", and "if", to connect statements.
Conjunction **(AND);** A conjunction or AND expression is based upon the logical AND function,
Disjunction **(OR);** another useful operator is the disjunction, Negation **(ROT);** the unary NOT operator,
Equality; the final simple connective is the equality connective. Implication **(IF-Then);** in development knowledge representations, one of the most useful logical connectives is the implication (IF: operator. The implication operator is used to formulate conditional statements, that is, statements of the form

IF antecedent
THEN consequent

Symbolic information encoded via (possibly compound) "antecedent" and "consequent" statement, together with the implication connective: form the basis for many AI implementations, including rule-base< systems.

15.5- The Hatching Operation - Takes two terms and tries to make them identical by instantiating the variables in both terms.
Matching, if it succeeds, results in the most genera: instantiation of variables.

Declarative Semantic - Defines whether a goal is true with respect to given program, and if it is true, for what instantiation of variables it is true.

Procedural Semantic - Is a procedure for satisfying a list of goals in the context of a given program. The procedure outputs the truth or falsify of the goal list and the corresponding instantiations of variables. The procedure automatically backtracks to examine alternatives.

List - The list is a simple data structure widely used in nonnumeric programming. A list is a sequence of any number of items such as Country, Region, PoliticalParty, Export. Such a list can be written i; prolog as:

[China, iraq, lybia, france, germany]

A list is a data structure that is either empty or consists of two parts: a head and a tail. The tail itself has to be a list.

Lists are handled in Prolog as a special case of binary trees.

For improved readability Prolog provides a special notation fo} lists, thus accepting lists written

[Item1, Item2, ...]

or

[Head:Tail]

or

[Item1, Item2, ...: Others]

List can be used to represent sets although there is ~ difference: the order of elements in a set does not matter while the order of items in a list does; also, the same object can occur repeatedly in a list. Still, the most common operations on lists are similar to those on sets. Among them are:

- Checking whether some object is an element of a list, which corresponds to checking for the set membership;
- Concatenation of two lists, obtaining a third list, which may correspond to the union of sets;
- Adding a new object to a list, or deleting some object from it.

15.6- Logic Diagramming - Action diagram is originally designed to be as easy to teach to end users as possible and to assist end user~ in applying fourth-generation languages. Most of the leading CAS! tools use action diagrams. They are useful for showing, and modifying, human agendas and procedures, and are used in the procedure boxes throughout this trilogy. They are particular~ useful because of the ease with which they can be expanded, contracted, and edited on the screen of a personal computer.

Decomposition diagrams enable a higher-level overview statement about a design to be successively decomposed into finer and fine} detail. They are sometimes referred to as function decomposition, process decomposition, and procedure decomposition diagrams.

15.7- Backtracing - In PLP, to establish whether an object satisfies a query is often a complicated process that involves 10gicaJ inference, exploring among alternatives and possibly backtracing. All this is done automatically by the prolog system and is, ir principle, hidden from the user.

The cut facility prevents backtracing. It is used both to improve the efficiency of programs and to enhance the expressive power of the language. Cut makes it possible to formulate mutually exclusive conclusion~ through rules of the form:

if Condition then **:** Conclusion1 **otherwise** Conclusion2

Cut makes it possible to introduce negation as failure **:** not Goal is defined through the failure of Goal. Simple objects in PLP are atoms, variables and numbers. Structured objects, or structures, are used to represent objects that have several components. Structures can be naturally pictured as trees. Prolog can be v1ewed as a language for processing trees.

15.8- , Basic Problem Solving Strategies, (State space) - Is a formalism for representing problems. State space is a directed graph whose nodes correspond to problem situations and arcs to possible moves. A particular problem is defined by a start node and a goal condition. A solution of the problem then corresponds to a path in the graph. Thus problem solving is reduced to searching for a path in a graph.

Optimization problems can be modelled by attaching costs to the arcs of a state space.

A search space, also called a problem space, is a space within which is contained the set of states of the problem being cons1dered, the operators or moves that describe transitions between states, and the specifications of the initial state from the solution process begins and the goal state which defines the end of the search for a solution. The various states in the space are also often referred to as nodes, especially where a search tree or graph is used to organize the search space. If each node or state in the represents or describes an actual configurations of the evolving solution, then the problem space is called a **state space.**

A search tree is a special case of a search graph. In a tree, nodes are very often repeated; that is, duplicates or copies of nodes appear as the tree unfolds. This is due to the fact that each node in a tree

has only one parent. These search operators are quite often defined in terms of rules in which the **IF** clauses, the lefthand sides of the rules, define the situation(s) that must obtain before the rule can be applied or fired. The right-hand sides of the rules, the THEN statements, define a set of actions to be taken in a given situation, that is, if the data for the problem at hand **matcb** the situation data given in the left-hand side of the rule.
There are two important ways in which rules can be used in a rule-based system; one is called **forward chaining** and the othe:r **backward chaining.**

15.9- Polit-Semantic Network - Semantic networks (or net) are verl good at expressing polit-knowledge about class inheritance properties, default and perspectives. A semantic network consists of entities and relationship between the entities. It is customary to represent a semantic network as a graph. In a semantic net, information is represented as a set of nodes connected to each other by a set of labelled arcs, which represent relationship among the nodes. Semantic networks can be easily implemented in PLP.
The use of graphical constructs to specify both numerical and symbolic relations among sets of entities is fundamental to man) knowledge representation approaches. In fact, the careful structuring of this construct is fundamental to many efficient schemes for knowledge manipulation.
A semantic network, or simply semantic net, is a labelled digraph used to describe relations (including properties) of objects, concepts, situations, or actions.
Semantic nets are ideal visualization tools, widely used in AI to provide a mechanism to "get started" in the development of E knowledge representation. Knowledge in a semantic net may be naturally organized to reflect hierarchies and enable inheritance E semantic net may be naturally organized to reflect hierarchies and enable inheritance. Because a semantic net representation denote~ relations as labelled arcs, an algorithm for reasoning using semantic nets may make relevant associations simply by matching observed (or stored) evidence with the graphical structure.

Frames - In frame representation, facts are clustered around objects. The objective of frames is to group common knowledge together. 'Objects' here means either a concrete polit-real object or a more abstract concept, such as a class of objects or even a situation. Good candidates for representation by frames are, for example, the typical meeting situation or game conflict situation. Such situations have, in general, some common stereotype structure that can be filled with details of a particular situation. A frame is data structure whose components are called slots. Slots have names and accommodate information of various kinds.

15.10- Searching - The concepts search used to formulate and solve problems. Search is the process of ski fully looking through the set of possible solutions to a problem so as to efficiently find an acceptable solution.
We shall introduce the concept of searching for solutions within a search space. A search space, often called a problem space, is an imaginary space that contains not only a solution to the problem under consideration, but also definitions of initial and final states; partial, incomplete, or wrong solutions; and other relevant information about the problem, such as the relations between states and the operators that are used to move from state to state. In order to make this more concrete, we need to talk about representation. Representation is the process of formulating or viewing a problem so it will be easy to solve.

15.10.1- Depth-first Search; Is easiest to program, but is susceptible to cycling. Two simple methods to prevent cycling are: limit the depth of search; test for repeated nodes. Depth-first iterative depending combines the desirable proprieties of depth first and breadth-first search.
Can also define as an uninformed graph searching strategy which searches the graph by exploring each possible path through it until either the required solution or a previously encountered node is encountered. The nodes are expanded in order of depth: with the deepest node expanded first and nodes of equal depth expanded in an arbitrary order. To prevent searching of an infinite path, a path bound is usually fixed and nodes below this depth are never generated, thus the strategy is neither guaranteed to produce the shortest path to the solution if one exists, nor to find a solution even if one exists.
To implement a depth-first search:

1- Form a one-element stack consisting of the root node.
2- Cycle through Steps a-c until the stack is empty.
  - Test the top element of the stack to see if it is a solution.
  - Exit with success if the tested element is a solution.

- If the first element is not a solution, add its children to the top of the stack.

3- Exit with failure if no solution is found.

15.10.2- Breadth-first Search; Breadth-first search attacks tree layer by layer - one layer at a time. That is, all the nodes at ~ given depth are examined to see if they might be solutions before any of them are expanded.
Breadth-first search is probably most effectively used when most of the solutions are at relatively shallow depths in the tree.
Implementation of the breadth-first strategy is more complicated as it requires maintaining the set of heuristic search. To implement a breadth-first search:

1- Form a one-element queue consisting of the root node.
2- Cycle through Steps a-c until the queue is empty.

a. Test the first queue element to see if it is a solution.
b. Exit with success if the tested element is a solution.
c. If the first element is not a solution, add its children to the back of the queue.

3- Exit with failure if no solution is founded.

15.10.3- Heuristic search; Heuristic Information - Can be used t< estimate how far a node is form a nearest goal node in the state space. The best-first heuristic principle guides the search process so as to always expand the node that is currently the most promising, according to the heuristic estimates.
Heuristic search is a technique for state space searching, with the state space represented as a graph. It uses domain-specific knowledge expressed as a numerical evaluation function which assigns a number to each node of the graph. At each stage of the search. Heuristic develops the tip node with the best numeric score. Tip nodes may be stored on an agenda in order of numeric score.
Both the depth-first and breath-first search routines are blind. They search for a solution by relying solely upon moving from one goal to the other without the use of "educated guesses." While this process may be fine for certain controlled situations where you a~ the programmer have information that tells you to use one method over the other, what a generalized AI program needs is a search procedure that is, on the average, superior to either of these two techniques. The only way to achieve such a search is to add heuristic capabilities.
Heuristics are simply rules that qualify the possibility that ~ search is proceeding in the correct direction.

15.10.4- The Hill-Climbing Search Technique - Hill-climbing is the name we give to a variation of depth-first search in which ~ heuristic evaluation function is used to estimate the distance remaining to be traversed from a given node to the goal node. In some searches, this conceptual distance can indeed be a measure of geographic distance between two points. For example in classic operations research (OR) problems. To implement a hill-climbing search:

1- Form a one-element stack consisting of the root node.
2- Cycle through Steps a-c until the stack is empty.
a. Test the top element of the stack to see if it is a solution.
b. Exit with success if the tested elemented is a solution.
c. If the first element is not a solution!

Sort the first element's children by the estimated remaining distance to the goal: and then add the children to the top of the stack.

15.10.5- AND/OR Graphs; AND/OR graphs are a suitable representation for problems that can be naturally decomposed into mutual independent sub problems. It naturally suits problems that are decomposable into independent sub problems. Game playing is an example of such problems.
Nodes of an AND/OR graph are of two types: AND nodes and OF nodes.
A concrete problem is defined by a start node and a gaol condition. A solution of a problem is represented by a solution graph.
Solving a problem, represented by an AND/OR graph, involves searching the graph. The depth-first

strategy searches the graph systematically and is easy to program. However, it may suffer from inefficiency due to combinatorial explosion.
Heuristic estimates can be introduced to estimate the difficults of problems, and the best-first heuristic principle can be used to guide the search. Implementing this strategy is more difficult.

15.11- Planning - In planning, available actions are represented in a way that enables explicit reasoning about their effect and their preconditions. This can be done by stating, for each action, its precondition, its add-list (relationships the action establishes) and delete-list (relationship the action destroys).
Mean-ends derivation of plans is based on finding actions that achieve given goals and enabling the preconditions for such action.

1- Goal protection - Is a mechanism that prevents planned destroying goals already achieved.
Means-ends planning involves search through the space of relevant actions. The usual search techniques therefore apply to planning as well: depth-first, breadth-first and best-first search.
To reduce search complexity, domain-specific knowledge can be used at several of means-ends planning, such as: which goal in the given goal-list should be attempted next; which action among the alternative actions should be tried first; heuristically estimating the difficulty of a goal-list in best-first search.

2- Goal regression - Is a process that determines what goals have to be true before an action, to ensure that given goals are true after the action. Planning with goal regression typically involves backward chaining of actions.

3- Hon-linear Planning - Recognize the fact that actions in plane need not be always totally ordered. Leaving the order unspecific whenever possible allows economical treatment of sets of equivalent permutations of actions.

15.12- Understanding of Natural Languages - The autonomous understanding of natural language (speech) is an active area of current research. Applications for this capability "keyboard less' document preparation, voice actuated machine control, and operator less information and reservation systems.
There are several levels at which natural language can be processed: phonetic, phonological, morphological, lexical, syntactic, semantic and pragmatic. For the description of subsets of natural language with a view to automatic analysis these levels are relatively well defined. Moreover, even if there are more or les~ significant interactions between these levels, it is generally~ agreed that they form a sequence in the order given above. This succession does not necessarily correspond to a psychological reality but rather to a functional scheme of usage current in NLI systems. The description given here is functional, and in an implementation these levels might be integrated or further decomposed for computational reasons.

The **phonetic level,** quite distant from the main theme of this book, is concerned with the composition of speech sounds. These Domains are closely tied to physics and acoustics. We can note tha1 research has been able to determine quite precisely which aspects 01 the speech correspond to which sounds.
The **phonological level** is also relatively infrequently integrated into current NLP systems, and concerns the way sounds combine t< form words., word groups and sentences. Again there exist sets 01 very precise rules describing how to pronounce a written tex1 (speech synthesis) or how to transcribe a sequence of sounds (speech recognition). Different levels of phonology are superimposed on each other, taking into account the pronunciation of individual syllables, features of linking or liaison between words introduction, and so on.

The **morphological level** deals with from of words in a sentence: verb conjugations, plurals of common nouns, and so or (inflectional morphology). This level also deals with more complex aspects such as the way words can be formed from other words: adjectives from nouns **(courage -> courageous),** nominalizations of verbs **(paint -> painter, painting)** and so on **(derivational morphology).** Typically in English and related languages derivations are formed by adding prefixes **(tidy -> untidy, force -> enforce)** or suffixes **(hope -> hopeless, hopeful).** Sometimes, derivations result in new words with quite distinct meanings **(revolve -> revolvert).** Sometimes included at this level is the treatment of components, where whole words are combined to form new meanings (a **blackboard** is a special type of blackboard). The morphological level is quite well delimited and can be handled with a set of rules and more or less short list of exceptions. The stud) of the formation of new words from existing ones is also quite advanced from the linguistic point of view, the relatively few NLI systems include rules for dealing with them since their syntactic and semantic behaviour (in comparison with that of the words from which they are formed) is often quite difficult to predict.

The set of words of a language is stored in what is known as the **lexicon.** The different morphological forms of the word are accessible via morphological rules, that is, inflected or derived forms can be analysed or generated forms are actually stored in the lexicon which will also indicate the information associated with their inflections (like tense of verbs, number of noun, and so on), as well as the root word from which they are derived. Alternatively, the lexicon may only list roots or stems, and there will be ar. associated set of procedures for getting from stems to full forms OI vice versa.

The **syntactic level** is concerned with the way words combine to form correct sentences. There are many formal ways of writing down the rules according to which sentences can be formed, corresponding to the informal descriptions familiar to us in 'grammar books for foreign languages.

The **semantic level** deals with the meanings of words, word groups and sentences. It is often closely connected to the **pragmatic level** which deals with the relationship between an utterance (or sentence) and its socio-cultural context. each word, at the lexical level, ha~ associated with it one or more 'internal' or semantic representation, which can vary from theory to theory.
To recap, here is a brief summary of the different levels discussed here:

- Information of sounds and words
  - phonetics
  - Phonology
  - Morphology
- Structure
  - Lexicon
  - rules of syntax and semantics
- Understanding
  - Semantics
  - Pragmatics

15.13- Machine Learning - The development of systems capable of learning involves the development or identification and implementation of algorithms ("learning algorithms") that enable other algorithms to improve their performance or to adapt on the basis of past experience or information.
Forms of learning include: learning by being told, learning from examples, learning by discovery. Learning concepts from examples is also called inductive learning.

A long term objective of CIPE-PLP is to complete development of an automatic discovery component that identifies novel situation~ and alternatives. The learning Component will consist of functions to:

- Select a "sentence" to be analyzed
- Note which actors reacted in different situations and what their reactions were
- Note what changes were made to the knowledge base after an event or events were processed.
- Note groups of events which appear together. These events could comprise a new script. Note where pre-established scripts were not followed.
- Note where recursion was used. Evaluate the advantages and disadvantages of using recursion in different situations.
- Analyze result from several "sentence" for commonalities or irregularities among them. Based on this analysis, test cases would be created.

An initial learning capability is currently being developed. Functions are being developed to analyze several scenarios to notice political process patterns, knowledge base changes, and polit-element reactions. For time being, test cases are manually created based on the results of this analysis.
Closely tied to intelligence is learning. In fact, intelligence cannot exist without the ability to learn because the principal advantage of learning is that it is the means of acquiring new knowledge. Learning allows you both to adapt to and use to your advantage various situations and events. Thus, the ability to learn is a powerful tool. It is surprise that many programmers want to create a program that can use this tool in the same way that people do. A program that could do this would be in theory the last program ever written because it could learn to perform various tasks on it~ own simply by being taught.

The development of systems capable of learning involves the development or identification and implementation of algorithm~ ("learning algorithms") that enable other algorithms to improve their performance or to adapt on the basis of past experience or information.
Forms of learning include: learning by being told, learning from examples, learning by discovery. Learning concepts from examples i~ also called inductive learning.

Learning from examples - Learning from examples involves: object!: and concepts, positive and negative examples of concept to be learned, matching between objects and concept descriptions Matching can be complicated and combinatorially complex.
The goal of learning from examples is to construct a formula in the concept description language that matches all the given positive examples and no negative example.

Inductive learning process - Involves search among possible concept descriptions, and is inherently combinatorial. To reduce computational complexity it can be heuristically guided.
During its construction, a **concept description** may be generalized 01 specialized. Normally, the final concept description is c generalization of the positive learning examples.

Purring decision trees - Is a powerful method for learning from noisy data.

Criteria for Success of Learning - Criteria for assessing the success of method of learning from examples include:

- Accuracy of learned concept description
- Transparency of learned concept description
- Computational efficiency in both constructing a concept description and in classifying new objects.

15.14- Logic Programming in use of Policy - Expert systems are computer programs that imitate a polit-expert. They contain polit-information (that is, a database) and a tool for understanding questions and finding the right answers to those questions in the database (that is, an inference engine). Polit-Logic programming has a built in structure for creating databases and has a ready-to-run inference engine. All you have to do is tell the program the rule~ to run by, and it will discover its own path to the appropriate information.
Polit-Logic programming incorporates a wide range of features, including a control mechanism, built-in predicates, and effective debugging facilities.
Logic-programming is based on a subset of the predicate calculus, including statements written as Horn clauses. The predicate calculus provides axioms and rules so that one can deduce new facts from other known facts. Statements expressed as Horn clauses allow a particularly mechanical method of proof called resolution.
A logic-based program consists of a series of axioms or facts, rules of inference, and a theorem or query to be proved. The output is true if the facts support the query, and false otherwise. PROLOG is the exemplar for logic programming languages.
[Polit] logic programming consists of defining relations and queering about relations. A program consists of clauses. These are of three types: facts, rules and questions. A relation can be specific by facts, simply stating the n-tuples of objects that satisfy the relation, or by stating rules about the relation. A procedure is a set of clauses about the same relation.

15.15- CPISE-PLP Expert Program - There are two important fundamental concepts to define at the outset: the strategic analyst and the strategic analysis problem. The strategic analyst is activity that may have learned one or more of the languages of the area. He ha~ undergone selected training.
The strategic analyst is supposed to do less than this: understand the dynamics and prospects of change in the area he watches; recognize signs of change, particularly threatening change; asses~ its significance; make projections; and carry out warning as appropriate.

Strategic Analysis is a rigorous cognitive process by which possible crucial realities of the future are first imagined and then modelled systematically to delineate their conditions, dynamics and potential outcomes, every effort being made to achieve realism and verisimilitude; with various inferential strategies procedurally employed to develop comparative probabilities; with the models and probabilities continuously reviewed and modified appropriately on the basis of new data;

A-major problem affecting the strategic analysis process is mindset, the tendency to perceive what you expect to happen rather than what actually happened. In many cases, the two perceptions coincide; but in some cases, they diverge dramatically, often with catastrophic results. Specifically, mindset leads an analyst to develop patterns of expectation; patterns which are sometimes biased by the analyst's point-of-view and background. Mindset also results because analysts have limited time to review all factors relating to their region of interest, an in many cases they are not able to perform detailed analysis of how events in their region might be related to events in other areas.

General awareness of these problems led us to realize there was a lack of adequate tools to properly assist analysts in overcoming mindset, and in exploring the ramifications of current events in a model of a geopolitical situation. CIPSE was developed to fill this gap. It aids analysts by exploring responses to specified events, suggesting alternatives for analysts, stimulating analytical thought, assisting in expansion and retention of strategic knowledge for analysts, and providing inputs for various reports. Specifically, CIPSE-PLP aids analysts by:

- Simulating sequences of events arising from event. This permits the analyst to quickly establish a wide range of plausible outcomes. One very important factor is that it is far easier for analysts to critique a large number of scenarios than to generate all of then themselves.
- Varying the initial assumptions, the analyst can explore alternative outcomes for a situation.
- Explaining the interactions among complex sequences of events.
- Providing a browsing capability for exploring a large knowledge base of facts about the world.
- Promoting the discovery of new insights through the use of the CIPSE-PLP automatic discovery component. An expert system is a program that behaves like an expert for some problem domain. It should be capable of explaining its decisions and the underlying reasoning. Often an expert system to be able to deal with uncertain and incomplete information.

It is convenient to .divide the development of an expert system into three main modules, as a knowledgebase, an inference engine, a user interface.

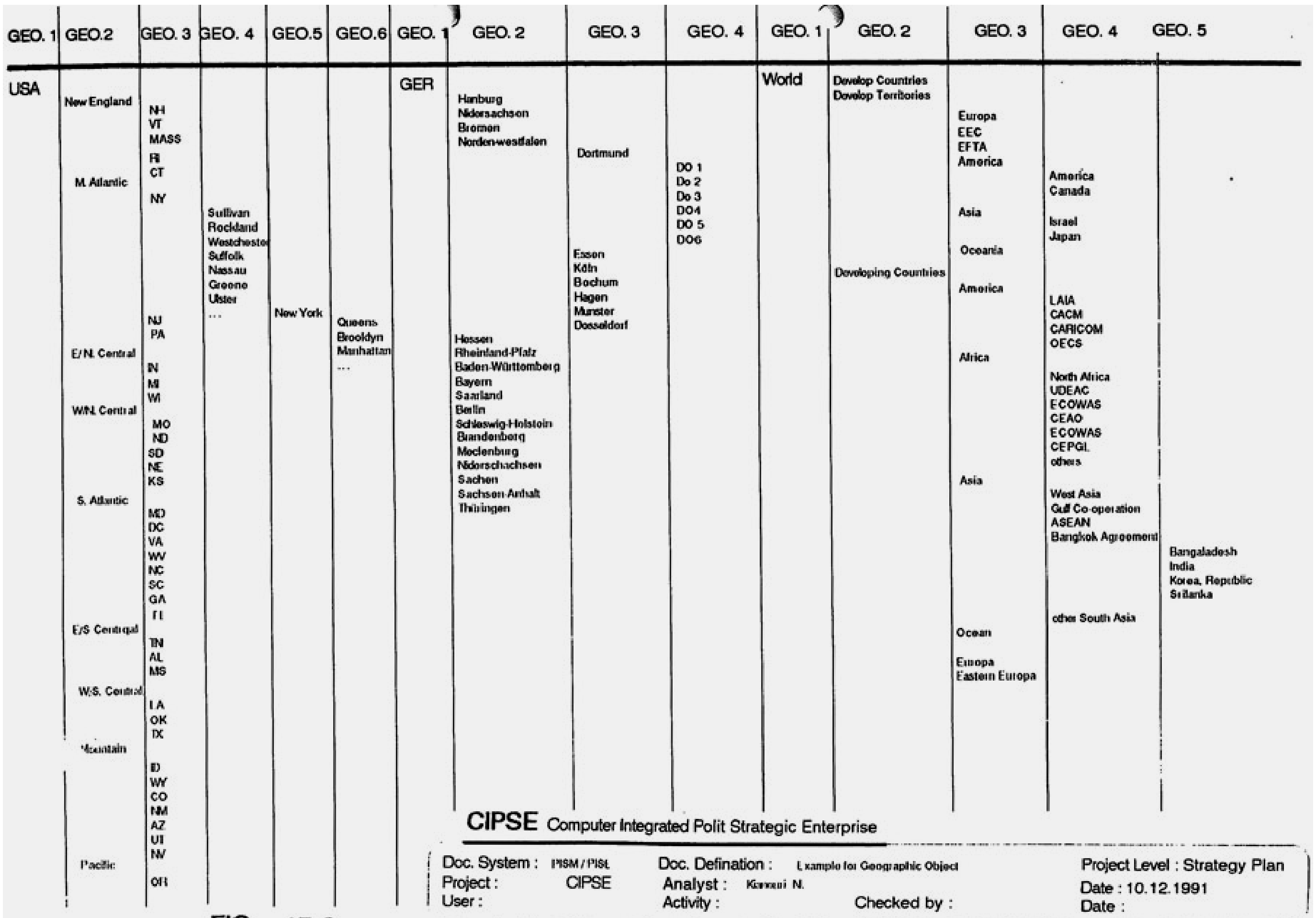
Typical functions required of an expert system are: solving problems in a given domain, explaining the problem-solving process, and handling uncertain and incomplete information.
It is convenient to view an expert system as consisting of two modules: a shell and a knowledgebase. A shell, in turn, consists of an inference mechanism and a user interface.

Building an expert system shell involves decisions regarding the knowledge representation formalism, the inference mechanism, the user interaction facility and the treatment of uncertainty.
If-then rules, or production rules, are the most common form of representing knowledge in expert systems.
Two basic ways of reasoning in rule-based systems are: backward and forward chaining.
CIPSE-PLP is a program special for analysis polit-purposes. The program is maked in base of semantic analysis of polit-process and elements in all aspect of foreign and intern policy.
PLP Expert Program has a big database of polit-information and possibility to run with many special procedures.
The central problem for design this program was that, how we car build a polit-expert system that can work near classic human polit-knowledge thinking process and not such as a abstract mathematical analysis of polit-processes.

15.16- CIPE/PLP Knowledge Base - The CIPE/PLP knowledge base contains symbolic descriptions of the various components (Geo, Elements, Process, Procedures, Time, Tactic, Strategy, rule) that re-present the simulated world, along with a small number of rules that guide the generation of plausible scenarios. All elements of the CIPE-PLP knowledge base are described by a data structure referred to as a unit. A unit is a frame that consists of a set of slots (e.g., military readiness, GNP, government stability). Slots are used to store values for the various characteristics describing the knowledge base elements. (e.g. country).

The knowledge base is structured into five taxonomic hierarchies (geographic, element, process, strategy, rule, and slot) that allow for inheritance of default values, and describe the interrelations among its elements.

Geographic - Geographic Areas represent real world geographic coordinate, continent, countries, bloc, region, and sub regions.



GEO. 1	GEO.2	GEO. 3	GEO. 4	GEO.5	GEO.6
USA					
	New England				
		NH VT MASS RI CT			
	M. Atlantic				
		NY			
			Sullivan Rockland Westchester Suffolk Nassau Greene Ulster ...		
				New York	
					Queens Brooklyn Manhattan ...
		NJ PA			
	E/N. Central				
		IN MI WI			
	W/N. Central				
		MO ND SD NE KS			
	S. Atlantic				
		MD DC VA WV NC SC GA FL			
	E/S Centrqal				
		TN AL MS			
	W/S. Central				
		LA OK TX			
	Mountain				
		ID WY CO NM AZ UT NV			
	Pacific				
		OR			

GEO. 1	GEO. 2	GEO. 3	GEO. 4
GER			
	Hanburg Nidersachsen Bremen Norden-westfalen		
		Dortmund	
			DO 1 Do 2 Do 3 DO4 DO 5 DO6
		Essen Köln Bochum Hagen Munster Dosseldorf	
	Hessen Rheinland-Pfalz Baden-Württemberg Bayern Saarland Berlin Schleswig-Holstein Brandenberg Meclenburg Niderschachsen Sachen Sachsen-Anhalt Thüringen		

GEO. 1	GEO. 2	GEO. 3	GEO. 4	GEO. 5
World	Develop Countries Develop Territories			
		Europa EEC EFTA America		
			America Canada	
		Asia		
			Israel Japan	
		Oceania		
	Developing Countries			
		America		
			LAIA CACM CARICOM OECS	
		Africa		
			North Africa UDEAC ECOWAS CEAO ECOWAS CEPGL others	
		Asia		
			West Asia Gulf Co-operation ASEAN Bangkok Agreement	
				Bangaladesh India Korea, Republic Srilanka
			other South Asia	
		Ocean		
		Europa Eastern Europa		

CIPSE Computer Integrated Polit Strategic Enterprise

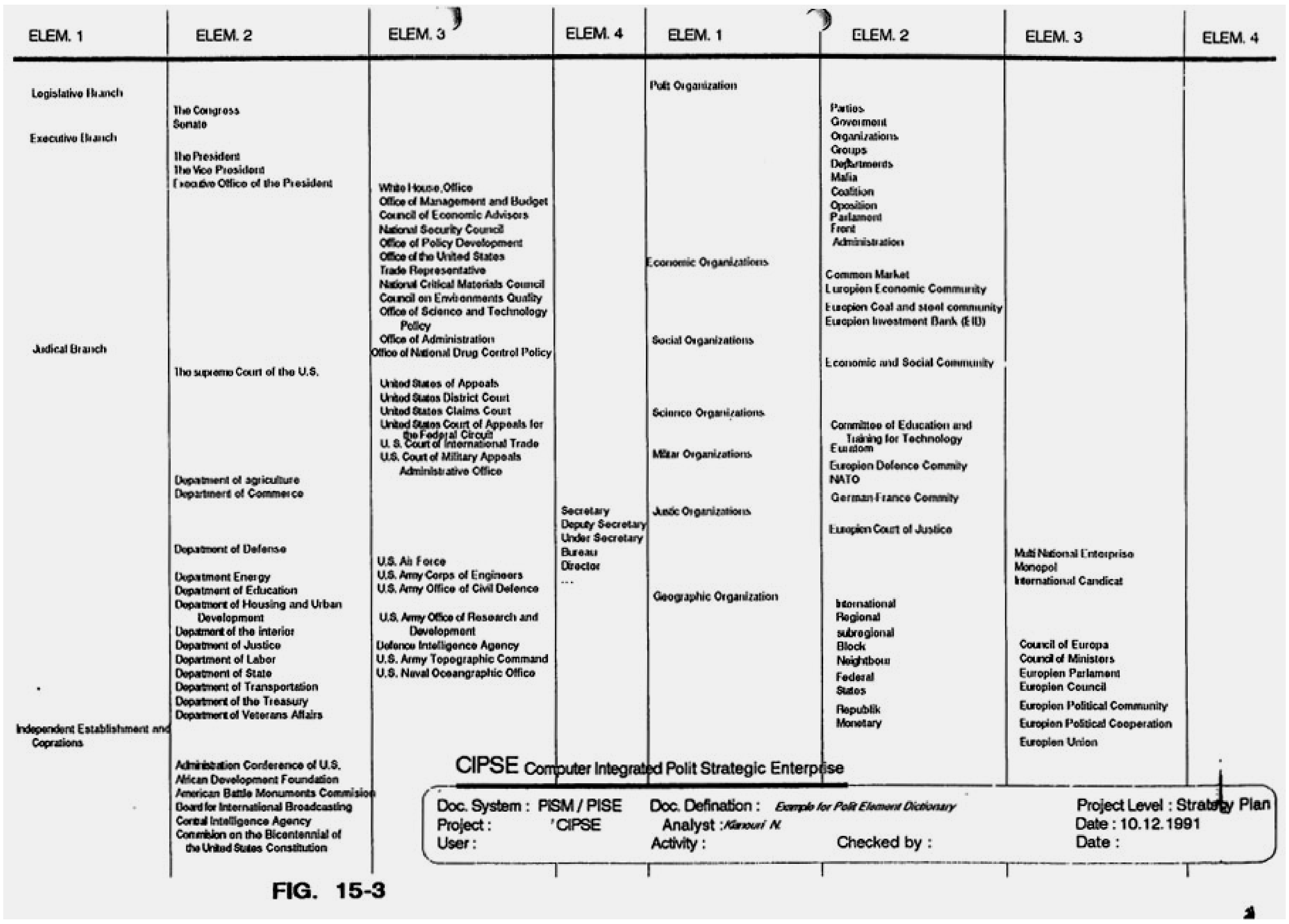
Doc. System : PISM / PISL	Doc. Defination : Example for Geographic Object		Project Level : Strategy Plan
Project : CIPSE	Analyst : Kamani N.		Date : 10.12.1991
User :	Activity :	Checked by :	Date :

FIG. 15-2

FIG.15.2. IS A DEMONSTRATION OF DIFFERENT POLIT GEOGRAPHIC AREAS. THE GEOGRAPHIC AREAS ARE A PART OF CIPSE/PLP PROGRAMMING.

Elements - Elements represent real world entities that could respond to events depending on the current situation. Elements can be thought of as having beliefs about themselves and other elements, and as having specific goals that they are actively trying to achieve. An element may be an individual (e.g., Margaret Thatcher, George Bush), a group (e.g, PLO, International Red Cross, British Government) a country (e. g., Indonesian, Libya), a class of elements (e. g., Western Actors), or a mechanical object (e. g., aircraft, ships).

A actor may have several sub actors which also represent independent decision makers. Sub actors of a country include its military, government, and civilian population.



ELEM. 1	ELEM. 2	ELEM. 3	ELEM. 4
Legislative Branch	The Congress Senate		
Executive Branch	The President The Vice President Executive Office of the President	White House,Office Office of Management and Budget Council of Economic Advisors National Security Council Office of Policy Development Office of the United States Trade Representative National Critical Materials Council Council on Environments Quality Office of Science and Technology Policy Office of Administration Office of National Drug Control Policy	
Judical Branch	The supreme Court of the U.S.	United States of Appeals United States District Court United States Claims Court United States Court of Appeals for the Federal Circuit U. S. Court of International Trade U.S. Court of Military Appeals Administrative Office	
	Depatment of agriculture Department of Commerce		Secretary Deputy Secretary Under Secretary Bureau Director ...
	Depatment of Defense	U.S. Air Force U.S. Army Corps of Engineers U.S. Army Office of Civil Defence	
	Depatment Energy Depatment of Education Depatment of Housing and Urban Development Depatment of the interior Depatment of Justice Department of Labor Depatment of State Department of Transportation Department of the Treasury Department of Veterans Affairs	U.S. Army Office of Research and Development Defence Intelligence Agency U.S. Army Topographic Command U.S. Naval Oceangraphic Office	
Independent Establishment and Coprations	Administration Conference of U.S. African Development Foundation American Battle Monuments Commision Board for International Broadcasting Central Intelligence Agency Commision on the Bicentennial of the United States Constitution		

ELEM. 1	ELEM. 2	ELEM. 3	ELEM. 4
Polit Organization	Parties Goverment Organizations Groups Departments Mafia Coalition Oposition Parlament Front Administration		
Economic Organizations	Common Market Europien Economic Community Europien Coal and steel community Europien Investment Bank (EIB)		
Social Organizations	Economic and Social Community		
Science Organizations	Committee of Education and Training for Technology Euratom		
Militar Organizations	Europien Defence Commity NATO German-France Commity		
Justic Organizations	Europien Court of Justice	Multi National Enterprise Monopol International Candicat	
Geographic Organization	International Regional subregional Block Neightbour Federal States Republik Monetary	Council of Europa Council of Ministers Europien Parlament Europien Council Europien Political Community Europien Political Cooperation Europien Union	

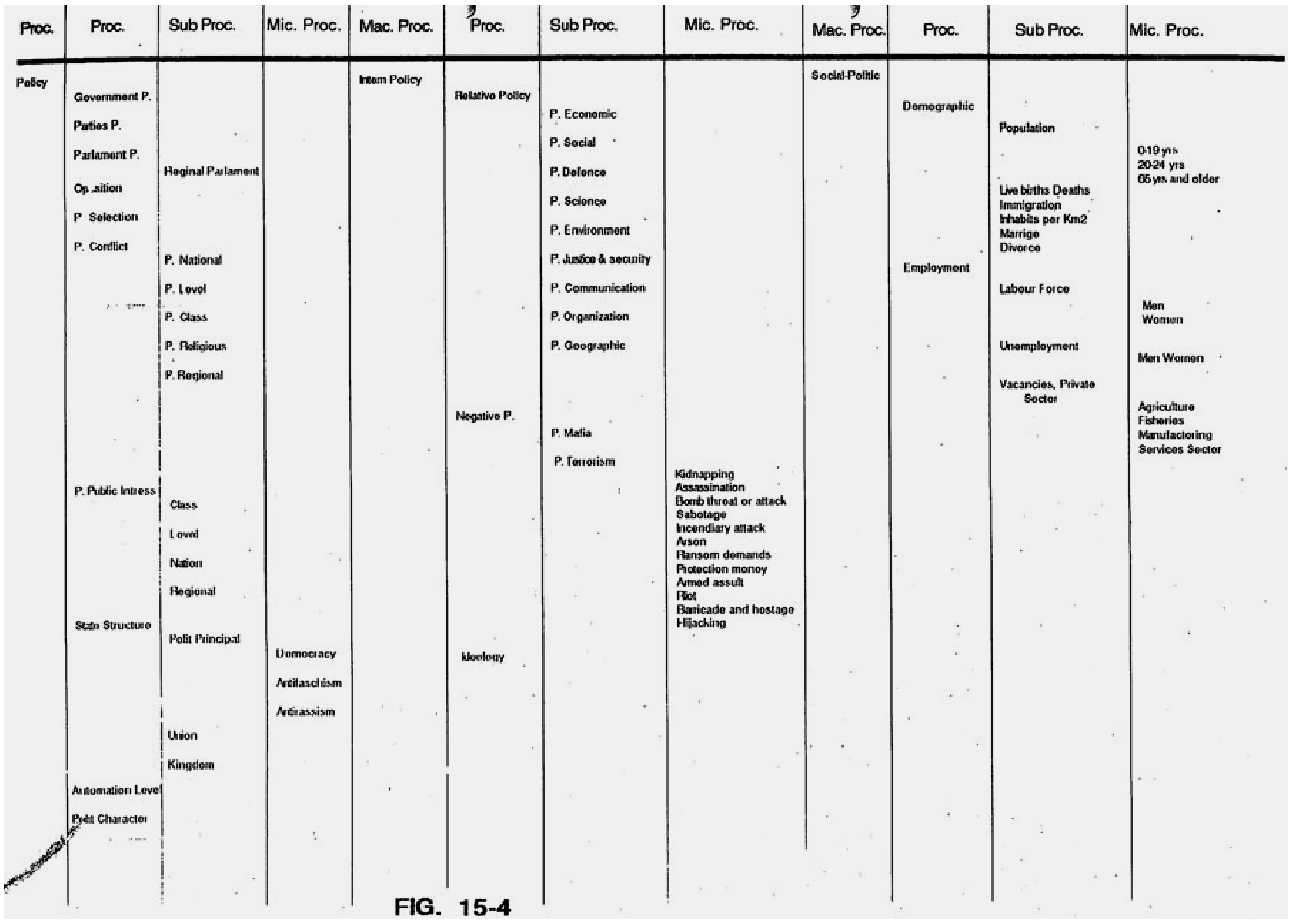
CIPSE Computer Integrated Polit Strategic Enterprise

Doc. System : PISM / PISE	Doc. Defination : *Example for Polit Element Dictionary*		Project Level : Strategy Plan
Project : 'CIPSE	Analyst : *Kanouni N.*		Date : 10.12.1991
User :	Activity :	Checked by :	Date :

FIG. 15-3

FIG.15.3. IS A DEMONSTRATION OF DIFFERENT POLIT ELEMENTS. THE ELEMENTS ARE A PART OF CIPSE/PLP PROGRAMMING.

Processes (Macro process, Process, Sub process, Microprocessor, and procedure) - An process corresponds to an action that occurred in the world at a particular time and place. A CPIE-PLP process would describe who was involved, what action was performed, how the action was performed, when it took place, why it took place, and who knows and believes the process took place. Processes are organized in a hierarchy based on action type (e.g., planning, attack, and announcement) as illustrated in figure 14-2.

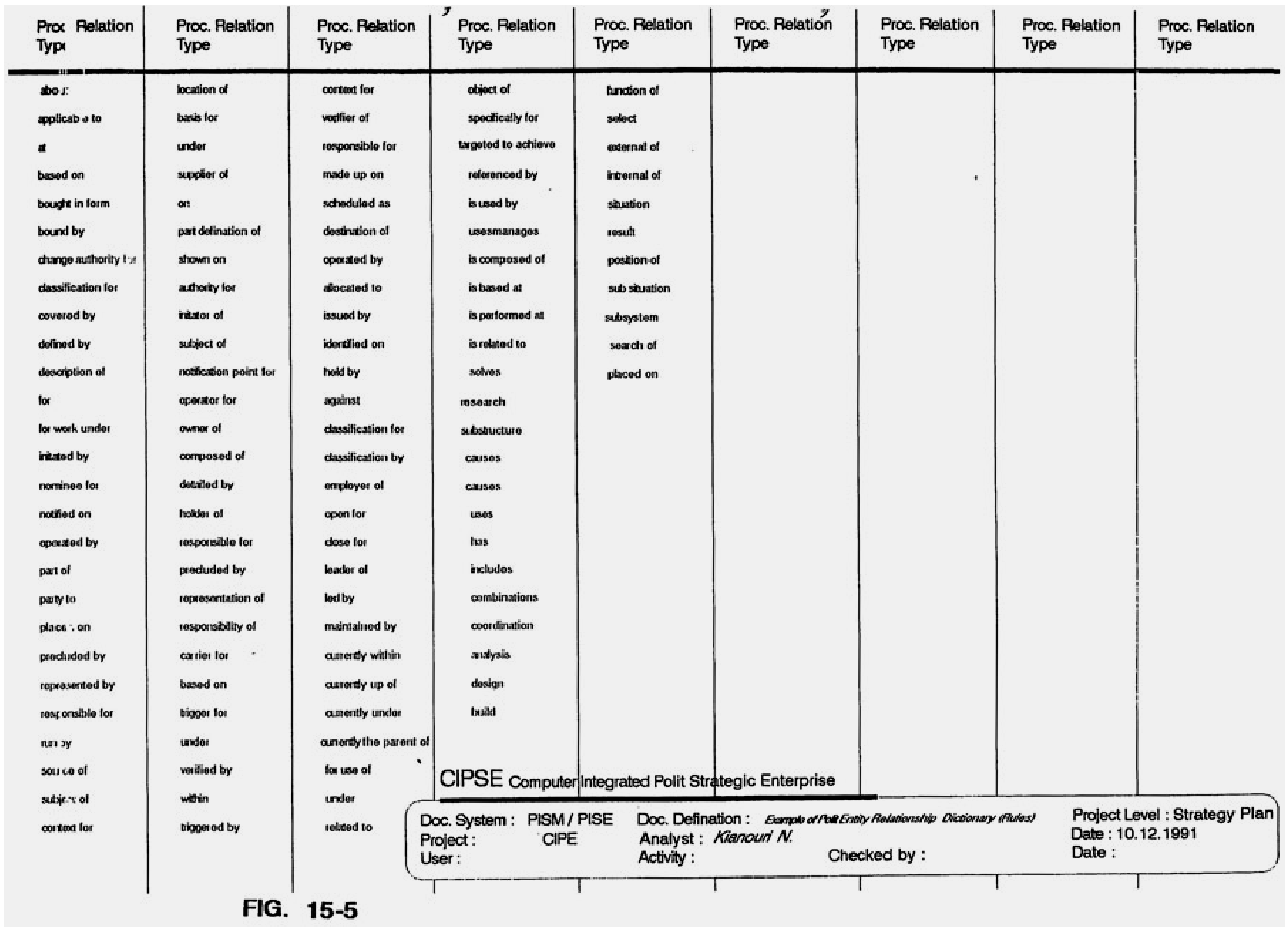


Proc.	Proc.	Sub Proc.	Mic. Proc.	Mac. Proc.	Proc.	Sub Proc.	Mic. Proc.	Mac. Proc.	Proc.	Sub Proc.	Mic. Proc.
Policy				Intern Policy				Social-Politic			
	Government P.				Relative Policy				Demographic		
	Parties P.					P. Economic				Population	
	Parlament P.					P. Social					0-19 yrs 20-24 yrs 65yrs and older
		Reginal Parlament				P. Defence					
	Op..sition					P. Science				Live births Deaths Immigration Inhabits per Km2 Marrige Divorce	
	P Selection					P. Environment					
	P. Conflict	P. National				P. Justice & security			Employment		
		P. Level				P. Communication				Labour Force	Men Women
		P. Class				P. Organization					
		P. Religious				P. Geographic				Unemployment	Men Women
		P. Regional								Vacancies, Private Sector	
					Negative P.	P. Mafia					Agriculture Fisheries Manufactoring Services Sector
						P. Terrorism	Kidnapping Assasination Bomb threat or attack Sabotage Incendiary attack Arson Ransom demands Protection money Armed assult Riot Barricade and hostage Hijacking				
	P. Public Intress	Class									
		Level									
		Nation									
		Regional									
	State Structure	Polit Principal	Democracy Antifaschism Antirassism		Ideology						
		Union									
		Kingdom									
	Automation Level										
	Polit Character										

FIG. 15-4

FIG.15.4. IS A DEMONSTRATION OF DIFFERENT POLIT PROCESSES. THE PROCESSES ARE A PART OF CIPSE/PLP PROGRAMMING.

Relation type - Represents relation between two or more geographician, element, event, and process.



Proc Relation Type	Proc. Relation Type	Proc. Relation Type	Proc. Relation Type	Proc. Relation Type	Proc. Relation Type	Proc. Relation Type	Proc. Relation Type	Proc. Relation Type
abo.:	location of	context for	object of	function of				
applicab e to	basis for	vedfier of	specifically for	select				
at	under	responsible for	targeted to achieve	external of				
based on	supplier of	made up on	referenced by	internal of				
bought in form	on	scheduled as	is used by	situation				
bound by	part defination of	destination of	usesmanages	result				
change authority for	shown on	operated by	is composed of	position-of				
classification for	authority for	allocated to	is based at	sub situation				
covered by	initator of	issued by	is performed at	subsystem				
defined by	subject of	identified on	is related to	search of				
description of	notification point for	hold by	solves	placed on				
for	operator for	against	research					
for work under	owner of	classification for	substructure					
initated by	composed of	classification by	causes					
nominee for	detailed by	employer of	causes					
notified on	holder of	open for	uses					
operated by	responsible for	close for	has					
part of	preduded by	leader of	includes					
party to	representation of	led by	combinations					
place . on	responsibility of	maintained by	coordination					
preduded by	carrier for	currently within	analysis					
represented by	based on	currently up of	design					
responsible for	trigger for	currently under	build					
run by	under	currently the parent of						
source of	verified by	for use of						
subject of	within	under						
context for	triggered by	related to						

CIPSE Computer Integrated Polit Strategic Enterprise

Doc. System :	PISM / PISE	Doc. Defination :	*Example of Polit Entity Relationship Dictionary (Rules)*		Project Level : Strategy Plan
Project :	CIPE	Analyst :	*Kianouri N.*		Date : 10.12.1991
User :		Activity :		Checked by :	Date :

FIG. 15-5

FIG.15.5. IS A DEMONSTRATION OF DIFFERENT POLIT RELATIONS. THE RELATIONS ARE A PART OF CIPSE/PLP PROGRAMMING.

Strategic Types - A Strategy represents an option for how an actor might respond to a particular situation. Specifically, it describes the set of processes (represented in the process hierarchy) that might take place if an actor chooses to respond to a process in a certain way. This response of elements largely corresponds to those elements attempting to achieve goals; in other words, a strategy describes the set of processes that would aid the element to achieve a goal.

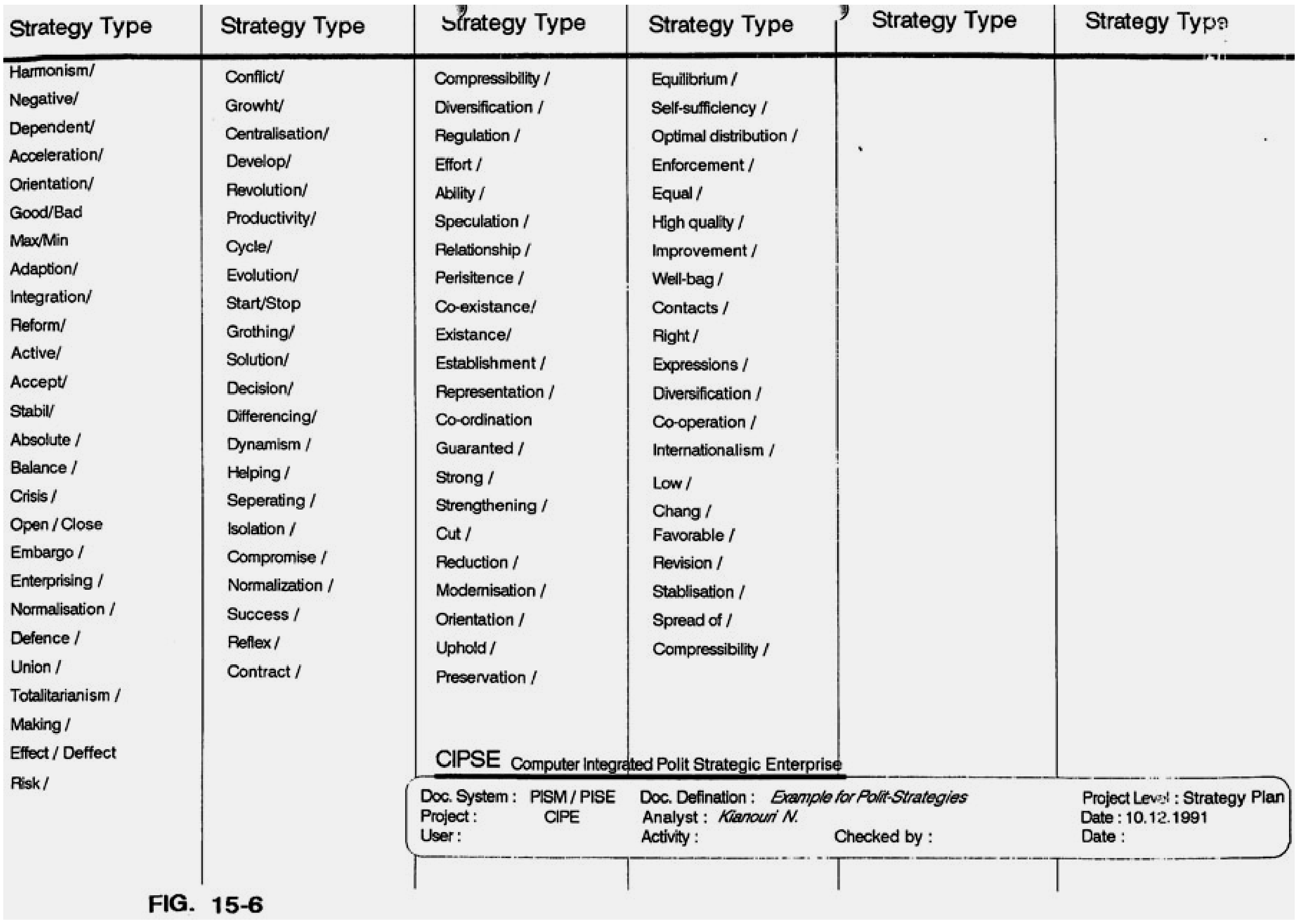
FIG.15.6. IS A DEMONSTRATION OF DIFFERENT POLIT STRATEGIE. THE STRATEGIES ARE A PART OF CIPSE/PLP PROGRAMMING.

15.17- CIPSE-PLP Project - Today, intelligence analysts spend most of their time determining the significance and implications of each new event as it occurs: in short, "strategists" still are largely reacting creatures. Few tools have been developed to assist ir exploring contingencies. The Polit-Logic Program (PLP) project which uses knowledge-based simulation to generate scenarios of world actors responding to given events.

The Polit-Logic Program (PLP) is a knowledge based simulator system for generating and analyzing alternative scenarios in geopolitical military situations. Given an event or process, PLP simulates the responses of various actors (countries, leaders, radical groups, etc) in the context specified by the particular scenario. These responses, in turn, generate new events which cause the process to repeat. A PLP simulation continues until no new events can be generated or until the termination criteria of the simulation are met.

PLP project is intended to become a tool that aids CIPSE intelligence analysts by evaluating the likely consequences of critical events. The basic need arises from the many documented human limitations, cognitive and institutional, which constrain strategic analysts from recognizing and anticipating potential] situations. For example, these limitations engender "mirror imaging", which prevents analysts from viewing a situation from the perspective of the countries being analyzed.

In sum, because of time and human limitations, analysts are not always able to review important factors relating to their region of interest; and in many cases they are not able to perform detailed analysis of how events in their region might be related to events in other areas. Yet our are more and more requires a global strategic perspective. Successful analysis will increasingly require labour-intensive efforts to generate and analyze contrasting outcomes of multiple situations.



Strategy Type	Strategy Type	Strategy Type	Strategy Type	Strategy Type	Strategy Type
Harmonism/	Conflict/	Compressibility /	Equilibrium /		
Negative/	Growht/	Diversification /	Self-sufficiency /		
Dependent/	Centralisation/	Regulation /	Optimal distribution /		
Acceleration/	Develop/	Effort /	Enforcement /		
Orientation/	Revolution/	Ability /	Equal /		
Good/Bad	Productivity/	Speculation /	High quality /		
Max/Min	Cycle/	Relationship /	Improvement /		
Adaption/	Evolution/	Perisitence /	Well-bag /		
Integration/	Start/Stop	Co-existance/	Contacts /		
Reform/	Grothing/	Existance/	Right /		
Active/	Solution/	Establishment /	Expressions /		
Accept/	Decision/	Representation /	Diversification /		
Stabil/	Differencing/	Co-ordination	Co-operation /		
Absolute /	Dynamism /	Guaranteed /	Internationalism /		
Balance /	Helping /	Strong /	Low /		
Crisis /	Seperating /	Strengthening /	Chang /		
Open / Close	Isolation /	Cut /	Favorable /		
Embargo /	Compromise /	Reduction /	Revision /		
Enterprising /	Normalization /	Modernisation /	Stablisation /		
Normalisation /	Success /	Orientation /	Spread of /		
Defence /	Reflex /	Uphold /	Compressibility /		
Union /	Contract /	Preservation /			
Totalitarianism /					
Making /					
Effect / Deffect					
Risk /					

CIPSE Computer Integrated Polit Strategic Enterprise

Doc. System : PISM / PISE	Doc. Defination : *Example for Polit-Strategies*		Project Level : Strategy Plan
Project : CIPE	Analyst : *Kianouri N.*		Date : 10.12.1991
User :	Activity :	Checked by :	Date :

FIG. 15-6

FIG.15.6. SHOWS TWO EXAMPLE OF POLIT LOGIC ANALYSIS, AND EXPERT RESULTAT AS A TEXT PRESENTATION.