Vorwort

Dieser Bericht enthält Erklärungen und Informationen über die verschiedenen Tätigkeiten, die ich seit Februar 1987 im Institut für Wasserwirtschaft der Universität Hannover (kurz IFW) geführt habe. Die erwähnten Punkten entsprechen mehr oder weniger der chronologischen Reihenfolge.

I) Es wurde erstens ein Experten-System zur Echt-Zeit-Steuerung untersucht.
Dies wurde von dem Diplomanten Andreas Neumann unter der Leitung von Dr.Ing Lothar Fuchs (IFW) und Dr. Müller (Institut für Mathematik der Universität Hannover) realisiert.
Anhang dieses Beispieles werden wir die Problematik erläutern, die zur Anwendung von wissensbasierten Systemen zur Kanalnetztsteuerung geführt hat.

II) Eines der wesentlichen Probleme bezüglich des Aufbaus solcher Steuerungssysteme betrifft ihre Flexibilität.
Die Frage lautet : kann man einfacherweise das System anwenden um nicht nur das ursprüngliche Netz (das Bremer Kanalnetz links der Weser) zu steuern, sondern ein beliebiges anderes ?
Modifikationen des Systemes mußten deshalb durchgefürt werden.
Die Organisation des geänderten Moduls wird beschrieben werden.

III) Die Tatsache, daß das System in Fortran geschrieben ist, bringt mit sich etliche Schwierigkeiten, deren Überwinden nur mit Einschränkungen der Möglichkeiten des Systems zu bezahlen ist.
Deshalb die Idee eine andere Sprache zu benutzen, um den Interpreter zu schreiben.
Zwei Sprachen haben sich als geeignet erwiesen insbesondere in der Entwicklungsphase eines Experten-Systemes, nämlich LISP und PROLOG.
Wir haben uns für die Sprache PROLOG entschieden.
Etliche Versuche, verschiedene Inferenz-Motoren in dieser Sprache zu erstellen, werden beschrieben und Beispiele gegeben.
- In der Tat können abhängig von dem zu lösenden Problem jeweils bestimmte geeignete Interpreter entworfen werden.-

Ein Prototyp für die Steuerung des Bremen Kanalnetzes.

Warnung :
Wegen zweier Gründe werden wir nicht bis ins Detail in die Struktur des von ersten Experten-Systems eingehen.

- Einerseits ist schon die Beschreibung der Struktur (zumindest der theoretischen Grundlage) anderswo vorhanden (A. Neumanns Diplom-Arbeit und Artikeln von L. Fuchs and co).
- Anderseits wird eine modifizierte Version des Experten-Systemes (an der wir unter anderen gearbeitet haben) gründlich beschrieben.

Da die beiden Versionen grundsätzlich ähnlich sind, brauchen wir nicht jeweils die selben Begriffe zu erklären.

I) **Warum** ein Experten-System für die Steuerung des Kanalnetzes im Bremen ?

I.1) Das Netz

Das Kanalnetz links der Weser entwässert ca. 1000 ha des Stadtgebietes, davon 470 ha undurchlässige Fläche.
Das Gesamtvolumen des Netzes umfaßt ca. 54 000 m^3, hinzu kommen 2 * 10 000 m^3 Volumen in zwei Rückhaltebecken.
Da das Einzugsgebiet sehr flach (Geländegefälle < 1/1000) ist, muß das gesamte Wasser, das in das Netz herein fließt, gepumpt werden.
Das Wasser kann entweder zu der Kläranlage oder zu dem Vorfluter gepumpt werden.

I.2) Die Steuerung

Früher war es so daß, um Überstau zu vermeiden, das Wasser so schnell wie möglich aus dem Netz gepumpt wurde.

Da die Kapazität der Kläranlage begrenzt ist, mußte ein großer Teil der Wasser-Menge ohne Reinigung in den Vorfluter entlastet werden.

Seit mehreren Jahren ist aber die Verminderung, der durch das Entlasten ungereinigten Wassers Umweltverschmutzung im Vordergrund getreten.

Die Stadtbehörden möchten das Ziel erreichen, daß 80% und mehr des gesamten Wassers, das in das Netz hereinfließt, durch die Kläranlage gereinigt werden kann.
Dies bedeutet, daß während eines Ereignisses ein großer Teil des Zuflußes innerhalb des Netzes erst gespeichert werden muß und dann nach Ende des Ereignisses zur Kläranlage geführt werden muß.

Die klassische Lösung besteht darin , daß man neue Rückhalte-becken aufzubauen versucht.
Diese Lösung wäre aber in unserem Fall völlig unrealistisch.
Die Kosten, die um genug Speicherkapazität zu schaffen, entstehen würden, überschreiten die finanziellen Möglichkeiten der Stadt.
Es wurde deshalb daran gedacht, die Speicherkapazitäten der Sammler durch eine angepaßte Steuerstrategie während des Ereignisses voll zu nutzen.

Durch diese Problematik sind schon die verantwortlichen Personen seit mehreren Jahren für die Echt-Zeit-Steuerung sensibilisiert. Steuerungsregel sind aus den Erfahrungen der Operateuren mit dem Netz entstanden worden, die in einem Steuerheft geschrieben sind.

Die Regel bestehen aus 2 Teilen :
- Ein Bedingungsteil
- Ein Konklusionsteil (anders genannt Aktionsteil).

Das Bedingungsteil entspricht dem Zustand des Netzes, im Moment wenn eine Steuerentscheidung getroffen werden soll.

Das Konklusionsteil bestimmt die zu treffende Entscheidung, wenn der Bedingungsteil erfüllt ist.

Die Grundidee, auf der die Zusammenarbeit zwischen dem Institut und der Stadt Bremen beruht, war erstens ein informatisches System zu bauen, das die in der Praxis benutzten Regeln enthält und sie automatisch bearbeiten kann.

Da die Steuerung 'optimiert' (oder zumindest überprüft) werden muß, sollte dasselbe System selbst seine ausgesuchte Strategie analysieren können und gegebenfalls automatisch neue 'bessere' Regeln erzeugen.

In anderen Worten ist das eingestrebte Ziel, ein Experten-System mit einem Lernmodul zu konstruieren.

II) Die Struktur eines Experten-Systemes.

Im allgemeinen besteht ein Expert-System aus 2 Teilen :
- der Interpreter oder die Schalle (the shell)
- die Wissensbasis (the knowledge base).

II.1) Die Wissensbasis

In der Wissensbasis ist das Wissen gespeichert, das speziell für den Anwendungsbereich zutrifft.
In unserem Fall enthält die Wissensbasis, das notwendige Wissen um jeweils eine Steuerungsentscheidung zu treffen.
Da dieses Wissen als eine Menge von (Produktions-) Regeln formuliert ist, enthält die Wissensbasis eine Produktionsbasis.
Diese Produktions-Regel sind in einem bestimmten Formalismus geschieben, den der Interpreter lesen und bearbeiten muß.

II.2) der Interpreter

Der Interpreter enthält drei wichtige Teile :
- Der Inferenz Motor
- Das Wissens-Erwerbs-Komponent
- Das Erklärungs-Komponent

II.2.1) Der Inferenz Motor

Die theoretischen Grundlagen stammen aus der Mathematik und Logik. Ein Inferenz Motor soll wie ein 'theorem demonstrator' funktionieren.
Ein Problem wird dann als eine Frage über die Gültigkeit eines Faktes betrachtet.

Wenn der Fakt als solcher in der Wissensbasis vorhanden ist, ist die Frage sofort gelöst.

Wenn der Fakt als solcher in der Wissensbasis nicht vorhanden ist, muß ein echter Inferenz Prozess gestartet werden.
Der Inferenz Motor bearbeitet dann die Wissensbasis, um jeweils aus den vorhandenen schon bekannten Fakten durch die Anwendung geeigneter Regeln Konklusionen zu ziehen oder Aktionen zu starten, die zur Antwort der ursprünglichen Frage führen können.
Die Auswahl, das Lesen und das Bearbeiten der angepassten Regeln wird als das Inferenz Prozess bezeichnet.

Die Gesetze der Logik besagen, daß falls der Bedingungsteil einer Regel erfüllt ist, muß dann der Konklusionsteil der Regel als gültig (wahr) angesehen werden.

Dieser Konklusionsteil ist eine Menge von neuen Fakten, der mit den dynamischen Fakten hineingetragen wird.

Die dynamischen Fakten sind kurzfristige Information, die nur für einen bestimmten Fall gültig sind. Es ist :
- der jeweilige Zustand des Netzes, bevor ein Inferenz Prozess

gestartet wird
- die nach einem Inferenz Prozess erschlossenen neuen Fakten.

Die dynamische Wissenbasis wird auch Arbeits-Gedächtnis (Working Memory) genannt, um die Trennung mit der dauernenden (langfristige) Wissensbasis (insbesondere die Menge der ProduktionsRegeln = Production Memory) zu betonen.

Grundsätzlich gibt es 2 Arten einen Inferenz Prozess zu leiten :
- der Prozess ist vorwärtsverkettend (forward chainig)
- der Prozess ist rückwärtsverkettend (backward chaining).

In einem vorwärtsverkettenden Prozess werden alle gültigen Regeln (deren Bedingungsteil erfüllt ist) gestartet und die neuen Fakten (die Aktionsteile) gespeichert.
Jeder neuer Fakt wird dann nachgeprüft, ob er eine Antwort der ursprünglichen Frage bietet.
- Falls ja wird das Inferenz Prozess unterbrochen (eine Lösung des Problemes ist gefunden) .
- Falls nein wird das Inferenz Prozess fortgesetzt (eine Lösung ist noch nicht gefunden). Das heißt, die nächste relevante Regel wird gefeuert.

In einem rückwärtsverkettenden Prozess werden nur die Regeln untersucht, deren Aktionsteil dem zu prüfenden Fakt entspricht.
Wenn eine solche Regel existiert, wird ihren Bedingungsteil untersucht.
Falls der Bedingungsteil verifiziert ist, wird das Problem als gelöst gelten (Eine Lösung ist gefunden) ;
falls der Bedingungsteil nicht verifiziert ist, wird eine andere Regel gesucht, derer Aktionsteil eine Antwort des Problems sein könnte und dann untersucht. Solcher Rückgang zu einer anderen Regel heißt Backtracking.

Falls das Experten-System in einer klassischen Hochsprache geschrieben wird, besteht der Inferenz Motor aus einer Reihe von Algorithmen.
Sie beschreiben wie die dynamischen Fakten gewonnen werden können.

In einer 'logischen' Sprache (wie zum Beispiel einem Prolog-Dialekt) wird
- entweder ein eingebauter Inferenz Motor in Aktion treten, wenn die Regeln direkt im Formalismus der Sprache geschrieben sind,
- oder es wird anhand Klausen (Regeln in dem Formalismus der Sprache) der Inferenz Prozess beschrieben.

Bemerkungen :
1) Der eingebaute Inferenz Motor eines Prolog-Dialektes ist rückwärtsverkettend.

2) Die Programmen, die die Funktionen des Interpreters übernehmen, werden als die Schale (Shell) des Experten-Systemes gekennzeichnet.
Wenn die Schale einen effizienten Inferenz Motor enthält, soll es möglich sein, jeweils sie mit einer anderen Wissenbasis (Production Memory und Working Memory) zu verkoppeln, um unterschiedliche Probleme zu lösen.

II.2.2) Die Erklärungskomponente

Das Wissen, das der Expert besitzt, muß formalisiert werden und angepasst werden, um überhaupt von dem Interpreter bearbeitet werden zu können.
Es muß denn bei der Aufbau der Wissensbasis jemand da sein,
- der einerseits den Interpreter kennt,
- der andererseits mit dem Expert zusammenarbeitet, um das Wissen zu fördern.

Dieser Person wird der 'knowledge engineer' genannt.

Es ist sehr schwierig zu urteilen, ob die erhaltene Wissensbasis vollständig ist , das heißt ob sie das gesamte Wissen, das vom Experten benutzt wird, enthält. Deswegen kann man nicht einfach den Ergebnissen, die vom computer abgegeben werden, ohnehin vertrauen.
Es muß immer parallell gespeichert werden, wie das System jene

Ergebnisse erreicht hat. Dies ist die Aufgabe der Erklärungs-Komponente.
Im Grunde genommen handelt es sich um die Speicherung aller Regeln, die für die gegebene Lösung benutzt worden sind.
Die Unterschiede zwischen den Experten-Systemen liegen in der jeweils unterschiedlichen Komplexität der Dialog-Komponente.

II.2.3) Die Erwerbskomponente

Wie schon erwähnt ist die Frage, wie das Wissensbasis konstruiert und modifiziert werden kann, entscheidend.

Ein erster Weg besteht darin, daß der Interpreter über Programme verfügt, die das Übertragen des Wissens vom Expert zur Machine erleichtern. Man spricht von direckter Erwerbung.
Das Ideale wäre, daß der Expert einfach in seinem Formalismus das Wissen liefert und dann die Erwerbskomponente die Übertragung in den internen Machine-Formalismus vollzieht. Leider solch eine Dialog-Schnittstelle, die so zu sagen die natürliche Sprache bearbeiten kann, bleibt eine der grossen ungelösten Fragen in der K.I.

Ein zweiter Weg besteht darin, daß der Interpreter selbst in der Lage ist, jeweils die vorgeschlagene Lösung zu analysieren und wenn eine Lücke aufgefunden wird, entsprechende Modifikationen der Wissensbasis durchzuführen.
Das Experten-System besitzt in dem Fall eine Lernkomponente. Wir werden nachher ausführlicher auf das Thema eingehen, wenn wir die Lernkomponente des Systems (für die Steuerung des Netzes) untersuchen und die entsprechenden Algorithmen beschreiben werden.

Organisation des modifizierten Systems

I) Einführung

Um die Steuerung des Kanalnetzes durchzuführen, braucht man :

- ein erstes Modul, das für jeden Steuerzeitschritt die Steuerentscheidungen aussucht,

- ein zweites Modul, das die Vorgänge innerhalb des Netzes berechnen kann (simulieren). Jene sind von den Werten der verschiedenen Variablentypen abhängig.

Nach der Systemen-Lehre gibt es 3 zu berucksichtigende Variablentypen :

- Die Zustandsvariablen : Durch ihre Werte erfährt man den jeweiligen Zustand des Netzes.
z.B Durchflußvariable, Abflußvariable, Wasserstandsvariable.

- Die Störvariablen : Es sind unkontrollierte Größe, die den künftigen Zustand des Netzes beeinflußen.
z.B Niederschlags-Intensitäten (räumlich und zeitlich verteilt).
Man kann auch die Betriebsfähigkeit eines Kontrollgerätes als Störvariable verstehen.

- Die Kontrollvariablen : Es sind kontrollierbare Größe, die den künftigen Zustand des Netzes beeinflußen.
z. B Leistungen der Steuergeräte.

Sei ein gegebener Zustand des Netzes und bestimmte Werte der Störvariablen,
die Steuerungs-Aufgabe besteht darin, die Werte der Kontroll-Variablen so zu bestimmen, daß der 'beste' Zustand des Netzes sich künftig ergeben wird.

Wie schon erwähnt, der Zustand eines Netzes wird durch Kriterien bewertet :
- Überflutungen, Überstauen (Menge in m^3),
- Entlastung ungereinigten Wassers (Menge in m^3),
- Energiekosten (Menge in kw.h oder DM).

Wegen der Störvariablen und der begrenzten Möglichkeiten der Steuergeräte ist das Netz nur partiell steuerbar.
- Ausgegangen von einem gegebenen Zustand kann man nicht einen beliebigen Zustand am nächsten Zeitschritt erreichen.
Oder um es praktisch an einem Beispiel auszudrücken, es ist unmöglich Überstauen zu vermeiden, wenn der Niederschlag eine gewisse Schwelle (abhängig wiederum von der Netzeskonfiguration) überschreitet hat. -

Genau so wichtig wie die Steuerbarkeit ist der Begriff der Beobachtbarkeit. Die Frage ist ob und in wie weit der Zustand des Netzes bekannt werden kann.

Insbesondere in der Entwicklungsphase ist ein Simulationsmodul unabdingbar, da man das Experten-System nicht während eines wirklichen Ereignisses testen kann.

Außerdem wenn auch im realen Ansatz das Entscheidungsmodul über Meßwerte verfügt, um den Zustand zu kennen, kann trotzdem Simulationswerte unvermeidlich sein.
- Es kann passieren, daß die Meßungen eine unvollständige Kenntnis des Zustandes liefern, weil die Variablen sehr schwer zu messen sind (Flußwerte, Überflutungsmengen,...) oder weil das jeweilige Meßapparat außer Betrieb geraten ist. -

Man sollte versuchen so weit wie möglich die Trennung zwischen dem Entscheidungsmodul und dem Simulationsmodul zu respektieren, um die Flexibilität des Systemes zu bewahren.
Flexibilität bedeutet hier :

- schnelle und einfache Übertragung auf ein anderes Netz,

- Möglichkeit, verschiedene Simulationsmodellen einzusetzen.

Die Komplexität des einzustellenden Simulationsmodells hängt von den Zielen (braucht man eine sehr genau Kenntnisse des Zustandes) und den Bedingungen der Steuerung (Rechenzeit, Verbindungen mit den anderen Verarbeitungsprogrammen) ab.

II) Das Entscheidungsmodul

II.1) Einführung

Während eines Ereignisses werden jeden Zeitschritt Steuerungs-entscheidungen getroffen.
Sie werden vom Entscheidungsmodul entsprechend dem Netzzustand getroffen.
Der nächste Netzzustand am Ende des letzten Steuerungs--Zeitschrittes wird vom Simulationsmodul berechnet.

Das Simulationsteil in unserem Fall beruht auf den Programmen HYSTEM\EXTRAN, die im Institut entwickelt worden sind.

Das Entscheidungsteil ist als wissensbasierte System konzipiert und wird deshalb 'Expertensystem für die Steuerung' genannt.

Wir haben oben schon erwähnt, wie Expertensyteme im allgemeinen organisiert werden.
Wir werden diesmal die bestimmte Organisation dieses Systemes beschreiben und die benutzte Terminologie (Namen der Dateien) erläutern.
Die Tatsache, daß ein Lernprozess vorhanden ist, hat zur Folge eine größere Komplexität der Struktur, da zwei Ebene in der Wissensbasis zu unterscheiden sind.

Wie oben erwähnt, gilt in einem wissensbasierten System gründlich die Trennunug :
- Interpreter oder Schale des Experten-system,
- Wissensbasis (Knowledge base).

Der Interpreter ist hier als eine Menge von Programmen, die in Fortran geschrieben sind, entworfen.
Die Wissensbasis ist als eine Menge von Dateien, die Fakten und Regeln enthalten.

II.2) Die Wissensbasis :

II.2.1) Erste Ebene : der Entscheidungs-prozess. Die Dateien und ihre Nützlichkeit.

α) Die ursprünglichen Produktionen sind erst 'in dem Formalismus des Experten' geschrieben und in einer Datei ('**PR1DAT**'-Produktions-Datei) gespeichert.
Das System liest sie ein, und bearbeitet sie.
Die Bearbeitung besteht aus zwei Etappen :

- Erst gibt es einige Überprüfungen.
. Ist der Formalismus respektiert ?
. Sind alle Variablen, die in den Regeln auftreten, bekannt und koddierbar ?
. Taucht nicht mehrere Male dieselbe Regel auf ?
...

- dann eine Koddierung.
Weil der Inferenz Motor als Fortran Programmen geschrieben ist, kann er nicht direkt unkoddierte Regeln behandeln.
Die Tabelle, in der die koddierten Regeln gespeichert werden, heißt 'Production Memory' (**PRDMEM**) - ProduktionsGedächtnis -.

Die Produktionen werden in unabhängigen Blocken zusammengesetzt. Jeder Block enthält die Produktionen, die anwendbar sind, um eine bestimmte Entscheidung zu treffen.
Eine solche Unterteilung ist wichtig, um die Lesbarkeit des 'Production-Memory' und die Schnelligkeit der Entscheidungs-Suche zu verbessern.

ß) Die Fakten beschreiben für jeden Entscheidungs-Zeitschritt den Zustand aller wichtigen Variablen. Für jede zu treffende Entscheidung muß eine Regel ausgesucht, derer Bedingungen den Fakten (dem Netzszustand) entspricht.
Diese Fakten werden zur Zeit auschließlich von der Simulation erzeugt und durch Kommunikationsprogrammen zum Experten-System liefert. Das Heißt in einer Datei gespeichert. Die Datei heißt

'Working Memory' (**WRKMEM**) - Arbeits-Gedächtnis -.
Die Aktionsteilen der angewendeten Regeln werden als neue Fakten wiederum in 'Working Memory' eingetragen.

Da während eines Ereignisses es kaum vorstellbar ist, daß jemand von außern Fakten einfügt, gibt es keine Dialog-Komponente.
Das hat zwei wichtige Folgen :

- Die Erklärungskomponente ist auf einem Protokoll reduziert.
Für jeden Schritt werden das 'Working Memory' am Anfang des Entscheidungsprozesses wie die Liste der während des Vorganges angewandten Regeln gespeichert (in einer Datei **AUSDAT** - Ausgabedatei-).

- Falls ein Teil der Fakten (aus irgendeinem Grund) nicht verfügbar ist, wäre es theoretisch möglich, daß keine Regel mehr anwendbar ist.
Um einen solchen Fall zu vermeiden, ist der Inferenz-Motor so entworfen, daß nicht die reine logische Inferenz implementiert ist, sondern was man 'fuzzy' Inferenz nennen kann (confer die Struktur des PATMAT (pattern matching) Programmes) .

II.2.2) Zweite Ebene : Der Lern-Prozess. Die Dateien und ihre Nützlichkeit.

Die Fähigkeit zu lernen bedeutet, daß das System nicht nur die Fakten bearbeiten kann, um neue Fakten (die Entscheidungen) zu finden, sondern daß er selbst die Regeln bearbeiten kann, um neue Regel zu finden.

Wie die Strategie-Suche auf der Anwendung von Regeln beruht, so auch der Lernprozess.
Die Regeln, die im Lernprozess eintreten, werden Meta-Regeln genannt, um sie von den üblichen Produktionen zu differenzieren.
Sie sind in einem Block zusammengesetzt und neben den üblichen Produktionen in dem 'Production Memory' gespeichert. - Sie werden in dem Block Nummer 0 gespeichert-

Die Meta-Regeln stellen eine Diagnose, jedesmal wenn Störungen im Netz beobachtet werden.

Möglichen Störungen sind zum Beispiel, Überstauen, Einstauen, Entlastung ungereinigten Wassers ...

Die Diagnose besagt :
- welches Steuergerät (oder welche Steuergeräte) für die Störungen verantwortlich gehalten werden sollen,
- die Höhe der Verantwortung durch eine 'Strafe',
- wie die Leistung des Gerätes geändert werden soll,
- auf welchem Zeitintervall ab dem letzten Zeitschritt Korrekturen durchgeführt werden sollen.

Jeden Zeitschritt also wird die Gültigkeit der ausgewählten Strategie überprüft. Das heißt es wird überprüft ob Meta-Regeln anwendbar sind. - Dafür wird genau die gleichen Prozesse wie in der Bearbeitung der einfachen Produktionen verfolgt. -

Erst nachdem eine Diagnose erforderlich ist, werden die Regeln, die verantwortlich für die 'falsche' Strategie sind, ausgesucht und mit Strafpunkten versehen.

Die Suche der zu bestrafenden Regel erfolgt durch einen speziellen Prozess, der im Interpreter fest geschrieben steht (Programm NEWVAL).
Dazu muß der Interpreter für jeden Zeitschritt, der untersucht werden muß, in der Lage sein, den Netzs-Zustand und die Liste der angewandten Regeln wieder zu finden.
Diese Information liegt in dem 'Decision Memory' (**DECMEM**) - dem Entscheidungs-Gedächtnis - .

Die Nummer jeder bestraften Regel wird mit der Höhe der Strafe gespeichert. Es werden auch gleichzeitig in der Datei die Werte von Zustands-Variablen geschrieben. Deswegen heißt die Datei 'PArameter-Werte-DATei' (**PAWDAT**).
Diese Werte der Zustands-Variablen sind wichtig für den Aufbau einer neuen Regel aus der alten bestraften Regel. Es erlaubt

anhand statistischer Verarbeitung einen mittleren Netzs-Zustand zu berechnen, wenn die alte Regel versagt hat. Dieser mittlere Netzs-Zustand wird als Basis für den Bedingungsteil der neuen Regel benutzt.

Am Ende des Lern-Prozesses wird das neue 'Production Memory' (mit den neu erzeugten Regeln), in der Datei **'PR2DAT'** in unkoddierter Form (direkt vom Expert lesbar) gespeichert.

II.3) Der Interpreter : Liste der Programme in ihrer zeitlichen Abfolge und ihre Funktionen.

Die Common-Bereiche :

Es gibt 3 Common Bereiche :
- **COMCON** Namen und Dimensionierung der Variablen, die von dem Entscheidungs-Modul benutzt werden.
- **SIMCON** Namen und Dimensionierung der Variablen, die von dem Simulations-Modul behandelt werden.
- **JUNCTI** Namen und Dimensionierung der Variablen, die speziell für die Kommunikationsprogrammen (zwischen Simulation und Steuerung) dienen.

Da wir die Trennung so weit wie möglich vollziehen wollen, wurde entschieden, daß die Module auf zwei verschiedenen Computern laufen müssen.
Der Austauch der Variablenwerte erfolgt durch Kommunikations-programmen.
Dieselbe Variablen können deshalb in den jeweiligen Modulen unterschiedliche Namen tragen. Die Übereinstimmung muß während der Kommunikation überprüft und bewahrt werden.

Das Hauptprogramm heißt **CONTROL1**.
Das ganze Entscheidungs-Modul wird als **CONTROL1.EXE** auf einer Diskette gespeichert, und durch den Befehl CONTROL1 aufgerufen.

Es folgt die Liste der Unterprogramme in der Reihenfolge ihres Aufrufs mit einer Erklärung ihrer Tätigkeiten :
- Die Source-code (xxx.FOR) sind in dem Verzeichnis STEUER\ SOU gespeichert.-

a) Die Initialisierungsprogramme :

Es wird durch das Unterprogramm **CONINI** die Initialisierung aller benötigten Variablen durchgeführt. **CONINI** ruft mehrere Unterprogramme auf.

CONINI\CONDIM (Control Dimensionning) : Dimensionierung und Standard-Werte der Steuervariablen.
CONINI\STEDIM (Steurungs-Dimensionierung) : Dimensionierung und Standard-Werte der Netzs-Variablen .

CONINI\CINDAT :
. Einlesen der Daten, die für die Kenntnis des Netzszustandes erhalten sind.
. Einlesen der Produktions-Datei und Einschreiben der koddierten Produktionen in das 'Production Memory'...

CONINI\INISIO : Programm für die Vorbereitung der Kommunikation mit dem Simulations-Modul. (Schnittstelle Initialisieren).

CONINI\INIKOM : Durch diese Subroutine werden festgestellt, welche Variablen und in welcher Reihenfolge zwischen dem Simulations-Modul und dem Entscheidungs-Modul ausgetauscht werden.

CONINI\CONNUM : Das Programm ordnet die entsprechenden Variablen in den Simulations- und Entscheidungs-Modulen zu.
- Die gleiche Variablen sind unterschiedlich numeriert in den beiden Modulen.-

Ein erster Lauf der Kommunikationsprogramme findet dann statt.

b) Die Kommunikationsprogramme.

Es gibt 2 Subroutinen, die zum Austausch der Information zwischen dem Simulationsmodul und dem Entscheidungsmodul dienen. Der Unterschied liegt in der Richtung des Informationsflußes.

- **RXIST** SIMULATION → CONTROL. Das Entscheidungs-Modul empfängt die Werte des Netzt-Zustandes für den jetzigen Zeitschritt.

- **TXSOLL** CONTROL → SIMULATION. Das Entscheidungs-Modul sendet die Soll-Werte an den Kontroll-Geräten für den nächsten Zeitschritt.

Zum Zweck des Austausches dienen spezielle Unterprogrammen :

- **SENBIB** Send-Bibliothek (Austausch control → simulation)
- **RECBIB** Receive-Bibliothek (Austausch simulation → control)
- **MELNAK** ob ein NAK (Non Aknowledgment Zeichen) empfangen worden ist.
- Falls ein NAK empfangen ist, bedeudet es, daß die Übertragung nicht korrekt durchgeführt worden ist und deswegen wiederholt werden muß.

c) Die Steuerungsprogramme :

Das Haupt-Unterprogramm heißt **STEUER.**
Während seines Laufes werden verschiedene Unterprogramme aufgerufen. Sie werden in der Reihefolge ihres Aufrufs gegeben und ihr Tätigkeit erläutert.

- **INWORK** (Input Working Memory):
Einschreiben des jetzigen Netzszustandes in das Working memory (Arbeitsgedächtnis).

- **INSEDM** (Insert Decision Memory):
Einschreiben des jetzigen Zutandes in das Decision Memory (Entscheidungsgedächtnis).
Diese Information erlaubt den Wiederaufbau des Netzzustandes für einen gegebenen Zeitschritt - wichtig um ein Lernschritt durchzuführen-.

- **PATMAT** (Pattern matching) :
Der Entscheidungsvorgang (über ein Steuergerät) beruht auf der Anwendung von bestimmten dem Zustand passenden Produktions--Regeln.
Im diesem Programm werden die passenden Regeln ausgewählt. Dieser Anpassungsprozess wird in Englisch Pattern Matching genannt.

Dieses Programm wird auch für den Bewertungsvorgang der ausgewählten Strategie gestartet.
Da die Bewertung der Strategie durch Regeln (die sogenannten Meta-Regeln) vollzogen ist, muß auch ein Patern-matching Prozess gestartet, um die Menge der anzuwendenden Meta-Regeln zu bestimmen.

- **NEWORK** (New Working Memory) :
Das Aktionsteil der gerade angewandten Produktionsregeln (oder Meta-Regeln) wird in das Working Memory eingetragen.

- **NEWVAL** (New Value) :
Wenn eine Strategie sich als unbefriedigend erwiesen hat, werden die Regeln, die für den Mangel verantwortlich gehalten sind, vom NEWVAL ausgesucht, mit den in der Meta-Regel vorgesehenen Strafpunkten versehen. Nummer der Regel, Strafe, und Werte der Zustands-Variablen von Bedeutung werden dann in PAWDAT gespeichert.

Nachdem das **STEUER** Unterprogramm gelaufen ist, werden die getroffenen Entscheidungen durch **TXSOLL** zu dem Simulations-Modul geschickt. (Es wird angenommen, daß die Soll-Werte für jedes Gerät eingestellt worden sind).

Die Steuerungs-Schleife wird dann sich wiederholen.
Die Steuerung wird abgebrochen :
- wenn 'Control C' oder 'Control E' oder 'Control T' von der Tastatur eingefügt wird,
- wenn das Ende der Simulation erreicht ist (TIME = TEND),
- wenn das Entscheidungsmodul eine End-Meldung ('92') bekommt.

d) Der Lernprozess :

- Der Lern-Prozess fängt während der Steuerungszeit an, wenn Meta-Regeln benutzt werden, um bestimmte unbefriedigende Netzs-Zustände zu entdecken, die auf eine schlechte Strategie hinweisen (programm PATMAT).
Die für die Situation verantwortlichen Regel werden ausgesucht und bestraft.

- Am Ende der Steuerzeit wird der Lernvorgang vollzogen.
Alle bestraften Regeln werden untersucht. Wenn die Summe der Strafpunkten eine gewisse Schwelle überschreitet, wird eine Änderung der Regel durchgeführt.
Diese Änderung wird im Programm **NEWPRO** (New Productions Memory) gemacht.
Das neue Bedingungsteil der neuen Regel entspricht dem mittleren Zustand der bedeutenden Variablen, jedesmal wenn die Regel bestraft worden ist.
Das neue Aktionsteil der neuen Regel wird aus dem alten berechnet. Die durchzuführenden Änderungen sind in der Meta-regel enthalten.

- Die neue Menge der Produktionen (Production Memory mit den neuen Regeln) wird in der Datei PR2DAT (in der unkoddierten Form = direkt lesbar) gepeichert durch das Programm **OUTPRO** (Output Productions).

Wie kann man praktisch das System laufen lassen ?

Da eine vollständige Trennung zwischen der Simulation und der Steuerung geplant ist, werden die Programm-Pakete auf zwei verschiedenen Disketten gespeichert.
Eine Diskette 'Simulation' enthält die Simulationsprogramme.
Die Diskette 'Steuerung' enthält das Experten-System mit den zugehörenden Dateien.

Experten-System und Dateien sind in dem 'Steuer/Exe' Verzeichnis gespeichert. Hier ist ihre Liste :

1) Das System

- **'CONTROL1.EXE'** : das Experten-System.
Man muß nur CONTROL1 durch die Tastatur eingeben, um das Programm laufen zu lassen.
Es ist aber notwendig, daß andere Dateien in dem Verzeichnis vorhanden sind, die für die Steuerung wichtige Information enthalten.

2) Die Dateien

- **'EINDAT'**: Das ist die Eingabe-Datei.
In dieser Datei steht geschrieben die Namen der Dateien, die vom Experten-System gelesen oder geändert werden sollen. Es sind :

- **AUSDAT** : die Ausgabe-Datei.
Es wird insbesondere das ganze Protokoll der Steuerung geschrieben.
Das heißt : für jeden Zeitschritt werden das 'Working Memory', die angewandten Regeln, gegebensfalls die angewandten Meta-Regeln und die Liste der bestraften Regeln geschrieben.

- **OUTPEG** : In der Datei werden jeden Zeitschritt die Wasserstände in den wichtigen Punkten des Netzes geschrieben.
- **OUTPUM** : In der Datei werden jeden Zeitschritt die Pumpen-Leistungen geschrieben.

- **PUMDAT.NET** : enthält die maximalen Leistungen der Pumpen.

- **CODDAT** : enthält die Liste alle Variablen, die für das Experten-System eine Bedeutung haben sollen, und ihre Kodierung. Die Variablen sind von Bedeutung, wenn es eine Möglichkeit besteht, daß sie im Schreiben von Regeln benutzt werden können.

- **PR1DAT** (**PRDBORB**) : enthält die Liste aller Produktionen und Meta-Produktionen am Anfang der Steuerung-Zeit.
- **PR2DAT** (**NPRDBORB**) : enthält die Liste der Produktionen (und Meta-Produktionen) am Ende der Steuerungs-Zeit, nachdem ein Lernschritt ('NEWPRO') durchgeführt worden ist.

- **PAWDAT** (**PARTAB**) : enthält die Liste der bestraften Produktionen, die Werte der Strafen und die Werte von für die Regel wichtigen Parametern.

- **KONDAT** : hat in unserem Falle keine Bedeutung.
Die Datei enthielt die monatlich 3 höhsten Spitz-Leistungen jeder Pumpe über einer Viertel-Stunde. In Bremen ist diese Information notwendig, um die Energie-Kosten zu berechnen.

Bemerkung :
Die Namen, die unter Klammern gegeben sind, sind die echten Namen der Dateien im Anwendungs-Fall, wenn sie von der allgemeinen Terminologie (erst gegeben) abweichen.