Claude A. Cruz

Plexus Systems
Nashua, New Hampshire

Claude Cruz received his B.S. in electrical and biomedical engineering from the University of Southern California, and a joint M.S. in these same subjects from the University of Illinois. Following graduate school, Mr. Cruz spent 11 years with IBM. The first 5 years of this were devoted to various circuit and systems design functions. This was followed by a transfer to the IBM Palo Alto, California, Scientific Center, where Mr. Cruz's initial work involved processor architecture and artificial intelligence (particularly expert systems technology). In 1981, Mr. Cruz founded a wide-ranging IBM neural-net project. This work covered neural-net theory, applications, and implementation techniques. In particular, it led to the creation of three generations of sophisticated tools for neural-net experimentation. These include a novel network "compiler", an interactive network debug environment, and a parallel digital network emulator. In May of 1987, Mr. Cruz left IBM to become a neural-net consultant, and to pave the way for founding a neural-net company (Plexus Systems).

KNOWLEDGE PROCESSING USING NEURAL NETWORKS

Abstract

Artificial neural networks (ANN's) are novel information processing mechanisms. Such networks are based on formal models of the function and organization of biological nerve nets. Neural nets possess several traits which make them an attractive substrate for artificial intelligence applications. They are an inherently parallel processing mechanism promising great speed of operation. In addition, most artificial neural nets are adaptive in that the behavior of the network may change over time as a function of its "experience" (cumulative operating history). Traditional AI involves the emulation of very high-level behaviors and cognitive mechanisms. Neural nets are a comparatively simple, low-level mechanism. Thus, applying them to complex tasks requires the resolution of a great many issues. These include problems of knowledge representation, definition and implementation of inference mechanisms, and control of the inference process. This presentation will identify some of the particular issues to be addressed, as well as some possible approaches and early results in ANN-based knowledge processing.

NEURAL NETWORKS

AND

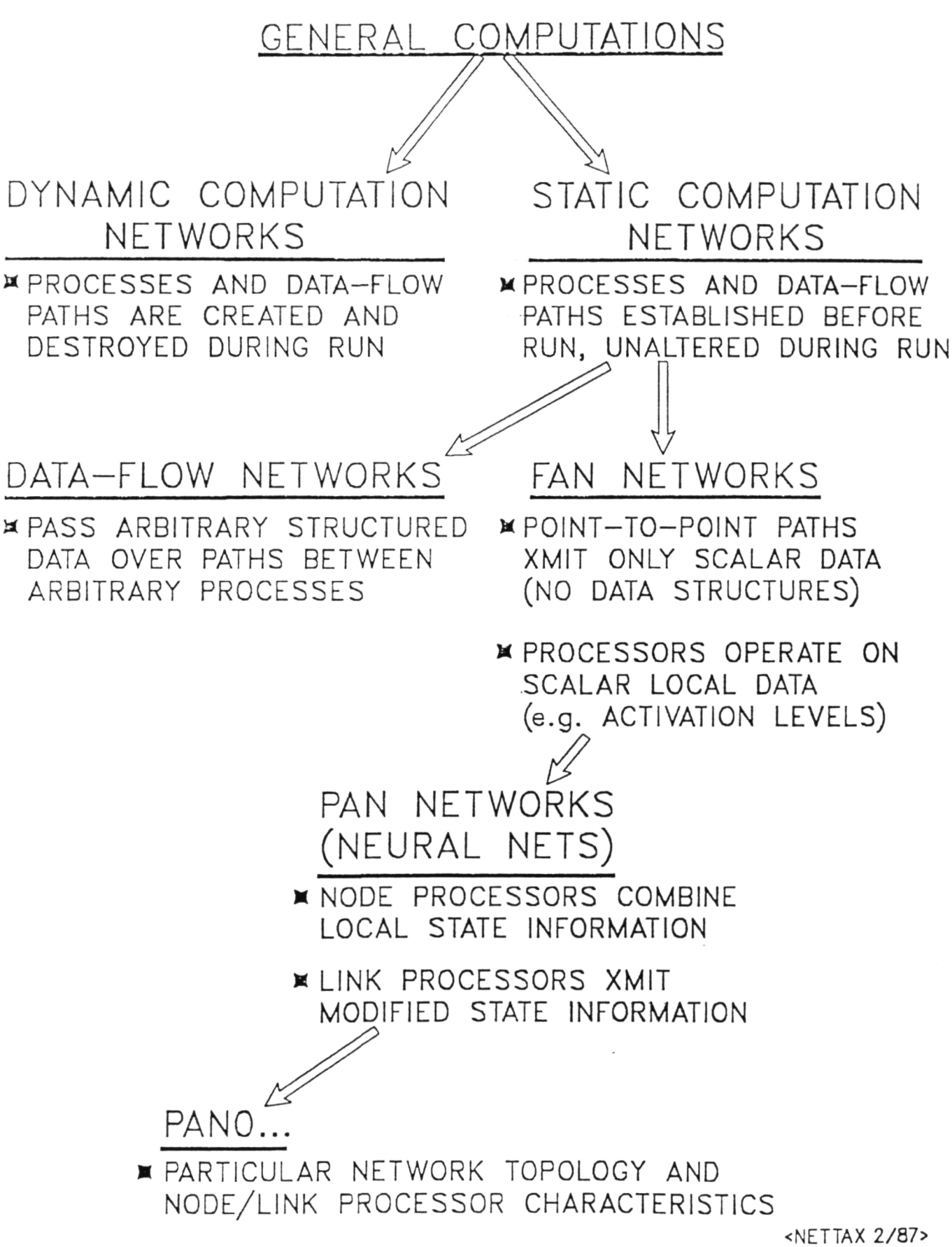
ARTIFICIAL INTELLIGENCE

Claude A. Cruz
Plexus Systems
10 Whitford Road
Nashua, NH 03062
(603) 595-2334

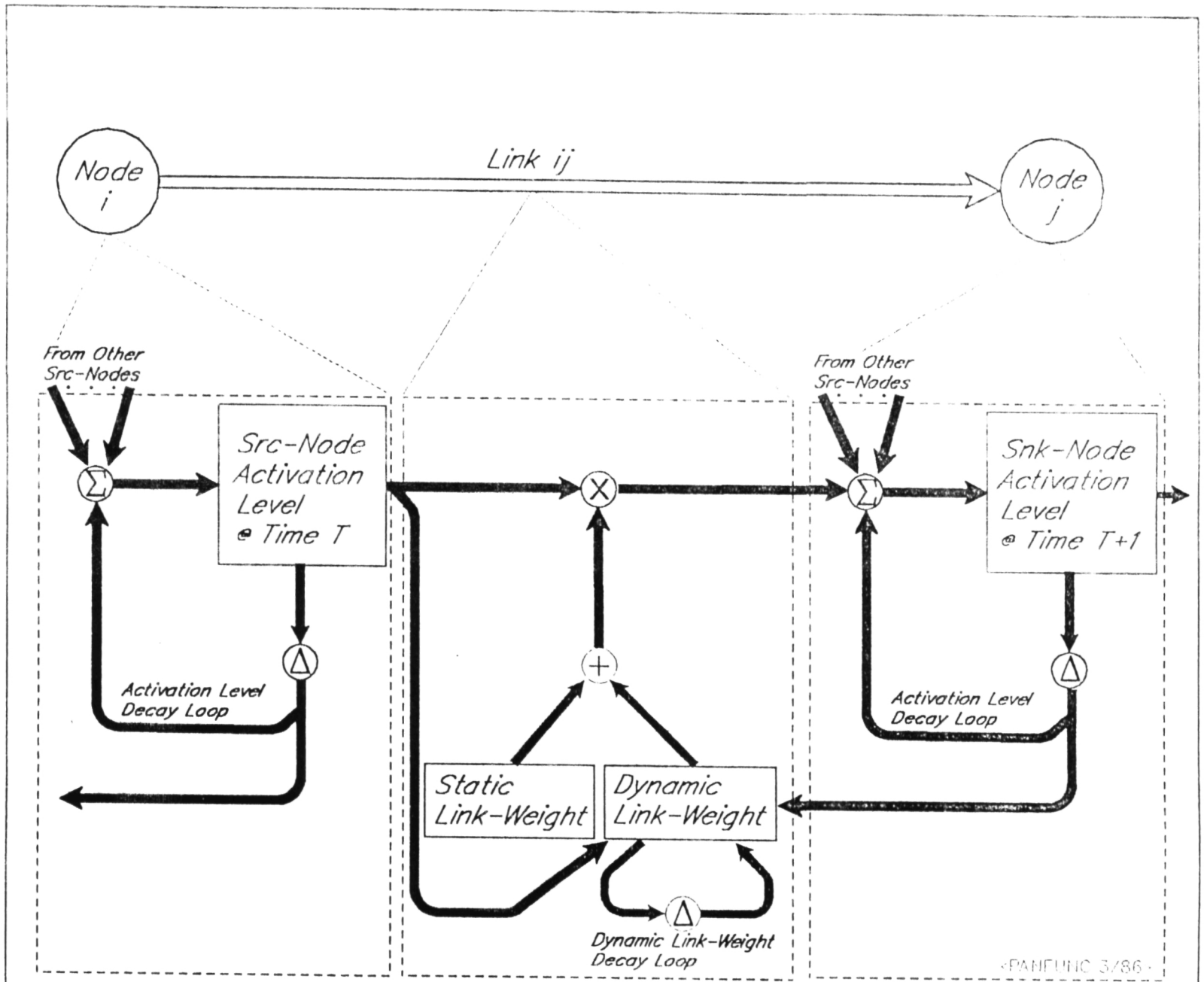
Overview

Project goal: explore *flow-of-activation networks* (FAN) as a new way to represent and process information:

- Inspired by function and organization of biological nerve nets.
- Use many simple processors to collectively perform operations (rather than one, or a few, processors operating independently).
- System performs "pattern processing" operations, rather than "symbolic" or "numeric" processing.
- FAN's can be *adaptive*; network behavior can change over time.
- Simple processors act as "smart memory"—system's processors and memory are combined.
- A FAN is a static network of "nodes" (processors) and "links" (communication paths). Acts like a "circuit" performing some function, rather than a sequence of procedure calls and data-structure instantiations.


GENERAL COMPUTATIONS
DYNAMIC COMPUTATION NETWORKS
PROCESSES AND DATA–FLOW PATHS ARE CREATED AND DESTROYED DURING RUN
STATIC COMPUTATION NETWORKS
PROCESSES AND DATA–FLOW PATHS ESTABLISHED BEFORE RUN, UNALTERED DURING RUN
DATA–FLOW NETWORKS
PASS ARBITRARY STRUCTURED DATA OVER PATHS BETWEEN ARBITRARY PROCESSES
FAN NETWORKS
POINT–TO–POINT PATHS XMIT ONLY SCALAR DATA (NO DATA STRUCTURES)
PROCESSORS OPERATE ON SCALAR LOCAL DATA (e.g. ACTIVATION LEVELS)
PAN NETWORKS (NEURAL NETS)
NODE PROCESSORS COMBINE LOCAL STATE INFORMATION
LINK PROCESSORS XMIT MODIFIED STATE INFORMATION
PANO...
PARTICULAR NETWORK TOPOLOGY AND NODE/LINK PROCESSOR CHARACTERISTICS

<NETTAX 2/87>


Node i
Link ij
Node j
From Other Src-Nodes
Src-Node Activation Level @ Time T
Activation Level Decay Loop
From Other Src-Nodes
Snk-Node Activation Level @ Time T+1
Activation Level Decay Loop
Static Link-Weight
Dynamic Link-Weight
Dynamic Link-Weight Decay Loop
<PANEUNIC 3/86>

Computing by flow-of-activation

- A non-von Neumann computing mechanism
- FAN attributes:
  - Highly parallel (fast)
  - Simple, uniform (pattern) processing
  - Can be adaptive ("learning" model)
- Basic bahavior:
  - Static net represents events and responses
  - Events "activate" nodes
  - Nodes drive other nodes by "flow-of-activation"
  - Active nodes trigger actions
  - Result: each event and each action processed simultaneously in parallel

Some Attributes of Natural Intelligence

- Purpose: to enable an organism to meet its needs (goals) in the face of a changing environment
- Learning is key; "hard-wired", "programmed" behavior limits organism's adaptiveness (like "brittle" programs)
- Nature of environment:
  - Contains regular "features" (entities and events)
  - Entities obey certain laws
  - Both features and laws exhibit some variability
- Intelligence capitalizes on these traits:
  - Learn to recognize features, and to associate significance ("meaning") with them
  - Predict attributes and behavior of features— build "world model" of environment
  - Use "fuzzy processing" to make time-varying best guesses about current contents of environment; seek adequate (not necessarily optimal) plan for meeting goals

System Embedded in Environment

- Cycle relating system and environment:

  1. Event occurs in environment (feature appears);

  2. System detects external event (internal event occurs);

  3. Internal event triggers system response;

  4. Response causes changes in environment (external events).

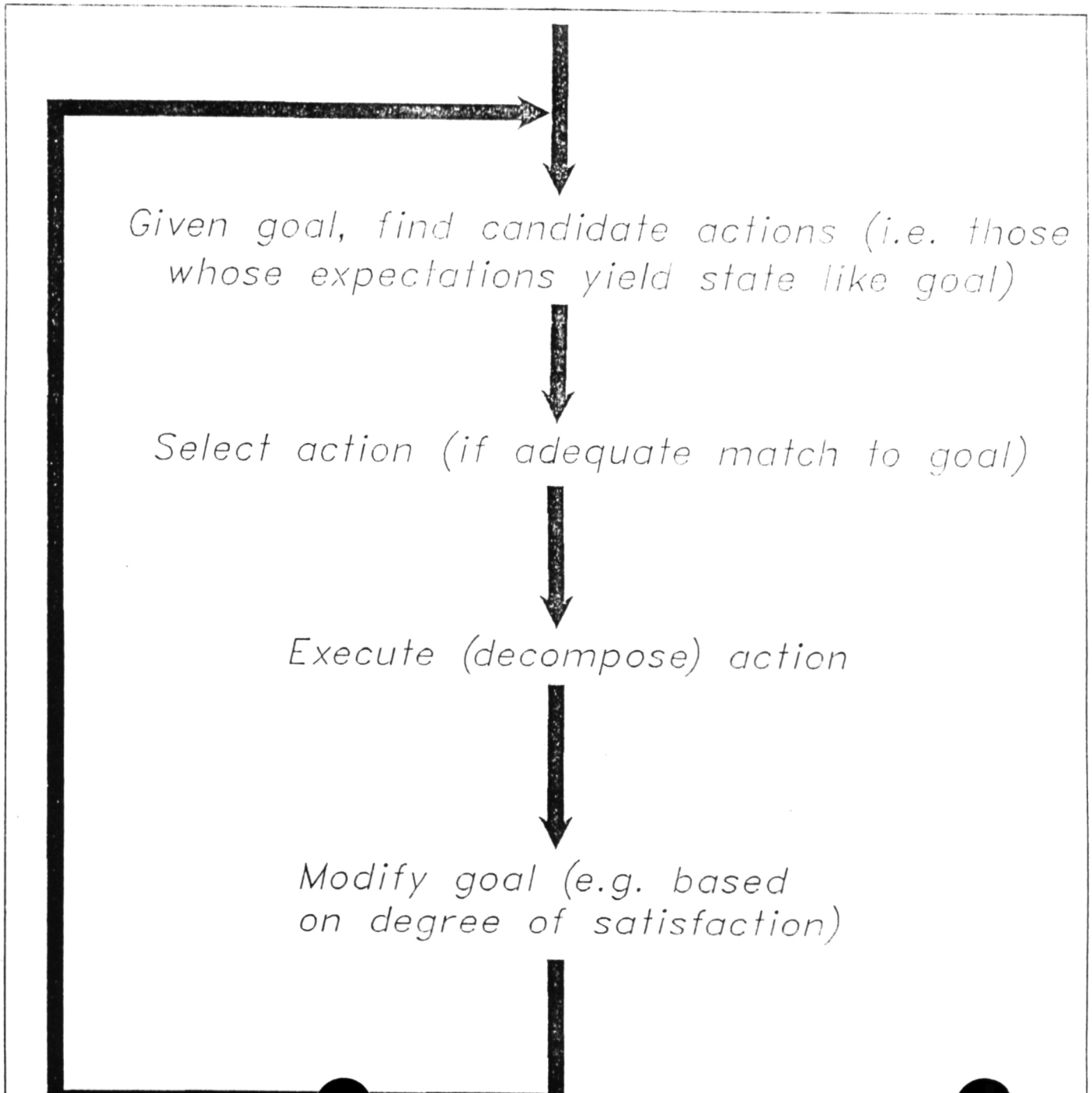
- System needs internal representation of external reality:

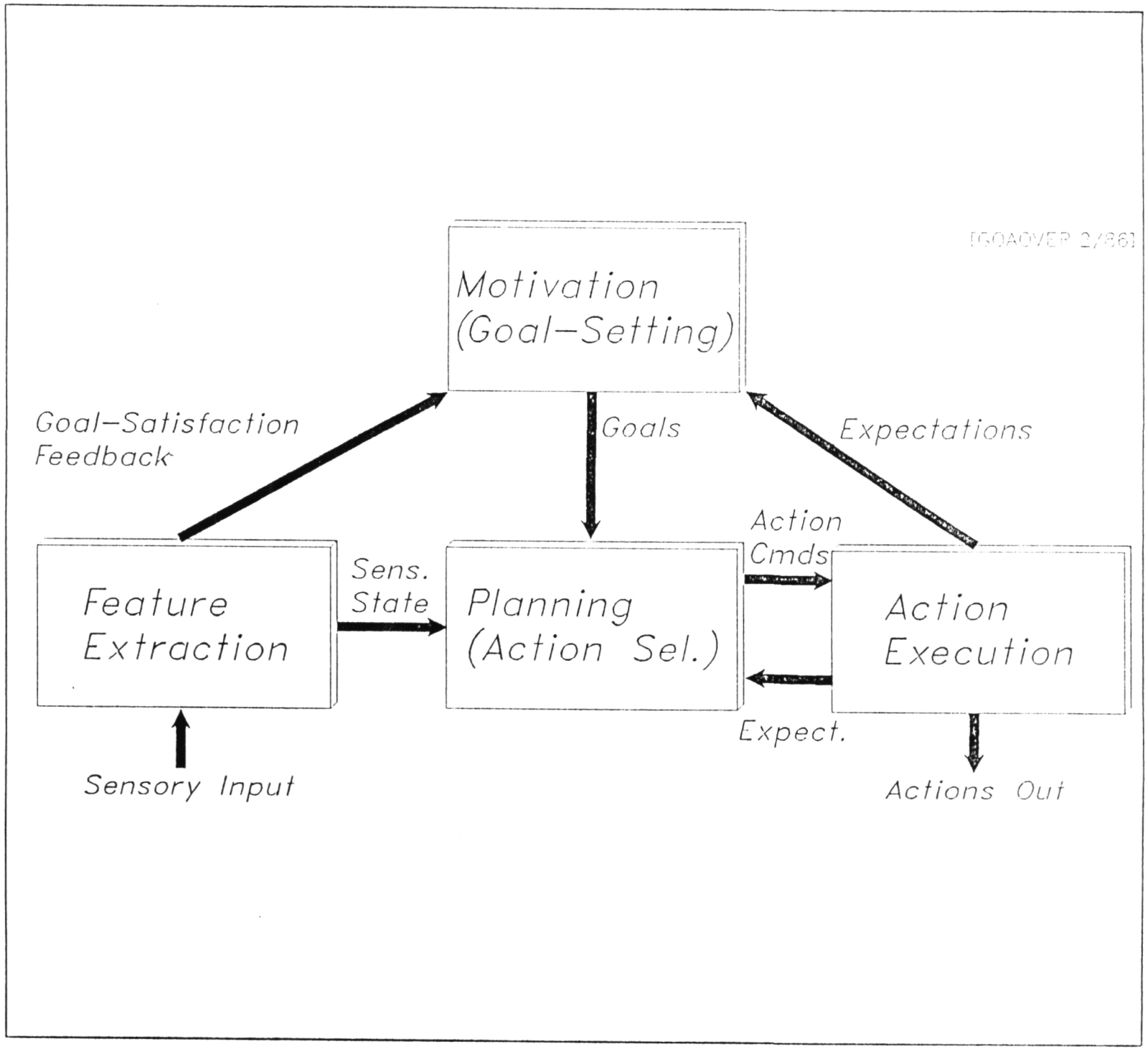
  - "Features" to represent external entities

  - "Relationships" to represent regularities between entities

  - "Operations" to embody responses to external events

  - "Rules" to associate specific responses with specific external or internal events


Given goal, find candidate actions (i.e. those whose expectations yield state like goal)
Select action (if adequate match to goal)
Execute (decompose) action
Modify goal (e.g. based on degree of satisfaction)


[GOAOVER 2/86]
Motivation
(Goal-Setting)
Goal-Satisfaction
Feedback
Goals
Expectations
Action
Cmds
Feature
Extraction
Sens.
State
Planning
(Action Sel.)
Action
Execution
Expect.
Sensory Input
Actions Out

Establishing Representations

- Purpose: set up equivalences (mappings) between external and internal entities:
  - External entities have regularity (content)
  - Presence of entity constitutes an event
  - Internal events produced by "detection" of external events (through fixed "transduction" mapping)

- Two basic schemes are possible:
  - *"Symbolic"* representation: assign a "symbol" (referent) to "stand for" (point to a description of) represented entity. This scheme separates the definition (meaning) of an entity from its referent, coupling the two through an (arbitrarily assigned) association. No relationship (e.g. transduction) required between content of external entity and form of internal representation pointed to by symbol.
  - *"Semantic"* representation: map the "transduction" of an external entity onto an internal "analog". This maps content of external event onto structure of internal representation. The mapping is fixed by transduction process (not arbitrary association). Mapping replaces use of referent (pointer).

Standard AI Techniques

Based on *"**symbolic processing**"*:

- Processing uses *descriptions* (formal models) , rather than *analogs*— incomplete information
- Processing based on *algorithms* (sets of explicit, detailed rules)— missing rules problematic, no generalization possible
- Data content (meaning) not used in processing; formal manipulation only— only programmer has knowledge, or "interpretation", of data's "meaning"
- Through above, processing uses only explicit relationships— cannot use implicit relationships, such as "similarity" between data
- Through above, "learning" (i.e. changes in knowledge-base) requires mediation by centralized "performance assessor"— inherently a serial, high-level function
- Processing is normally *precise*, as are symbols— no role for ambiguity, like that in fuzzy sets or approximate reasoning methods

- Control/decision-making normally centralized—parallel processing difficult, therefore slow

Intelligence via Artificial Neural Systems

Based on "*semantic processing*":

- Processing based on *analogs* of entities being reasoned about- – acquired through learning or detailed formal specification, thus can contain complete information

- Processing based on structure (contents) of knowledge-entities (KE's), plus connections (relationships) between them— inter-KE connections cause "rule-like" behavior. Generalization possible through "approximate" satisfaction of rules

- Data content (therefore structure) constrains inferences a KE can participate in— uniquely determines "interpretation" of KE's "meaning". No formal manipulation, just inference via "flow-of-activation"

- No implicit relationships, just explicit ones (through inter-KE connections)— "similar" KEs have similar (largely-shared) structure

- "Learning" occurs as local, low-level function (through change in inter-KE coupling)— no central

"performance assessor" needed (though one may be used; e.g. an attention mechanism)

- Both processing and KE's are normally imprecise (but can be made arbitrarily precise)— approximate reasoning is the norm

- Decisions made through locally-controlled flow-of-activation— processing is inherently parallel and fast (but can be made serial)

What Are Knowledge Representation Networks (KRN)?

- An attempt to emulate some biological information processing capabilities ("intelligence"):
  - Process multi-modal input (sensory processing)
  - Produce flexible output activity (distributed motor control)
  - Intelligent decision-making (cognitive functions; adaptive planning using a learned "world model")
- Assumption: choice of low-level implementation medium (FAN) is important:
  - Must encompass above three types of functions
  - Underlies important functional capabilities (associative storage, learning, distributed processing and memory, etc.)
  - Draw insight from biological organizational principles

KREV0287

KRN ARCHITECTURE

- Basic **knowledge units** (**features**, **operations** and **relationships**) are represented by **KRN networks**
- **Inferences** are performed through **interactions** between **knowledge units**
- Underlying **KRN has static structure** (no additions or deletions of nodes or links during inferences)
- Channelled flow of activation creates **dynamic knowledge structures** as subsets of overall KRN (like instantiation of variables)
- **PAN is ideal implementation medium** for KRN (parallel network, flow of activation, learning model)

KRN Architecture

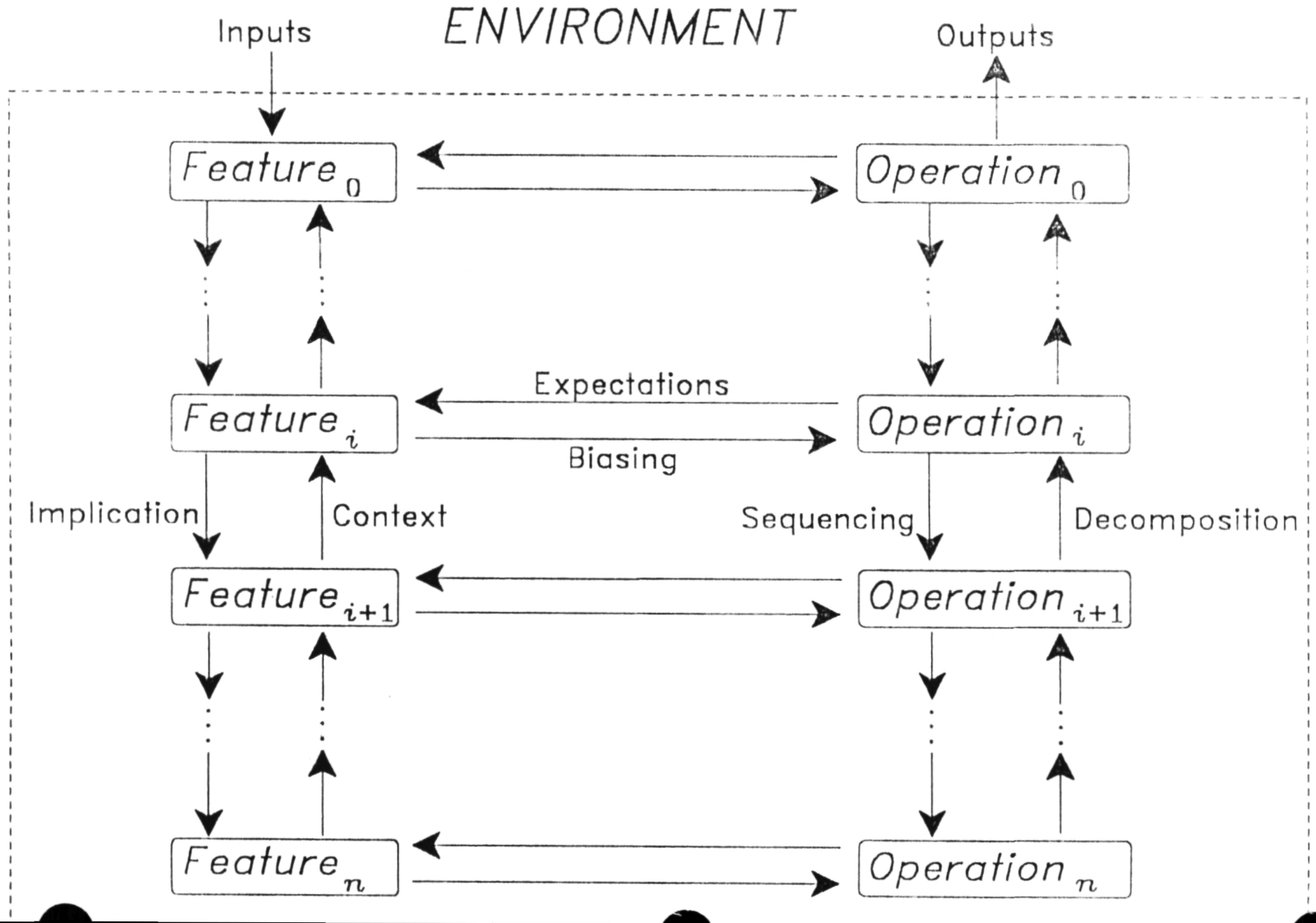
- Overall structure of KRN net— a "feature" hierarchy cross-coupled with an "operation" hierarchy (like advanced nervous systems):

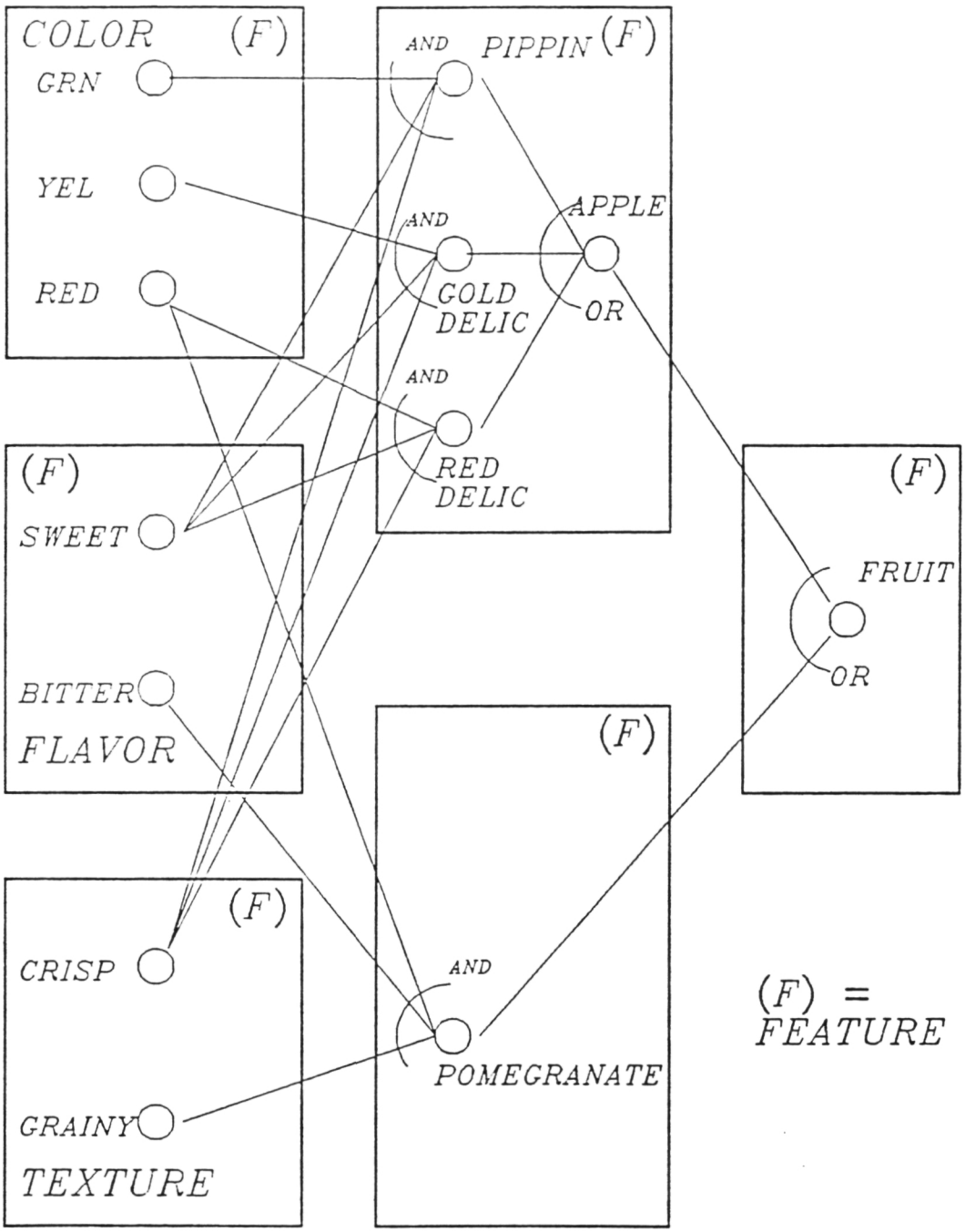
  - Feature hierarchy represents events and entities which the KRN net can recognize. Conjunctions of features define higher-level (more complex) features.

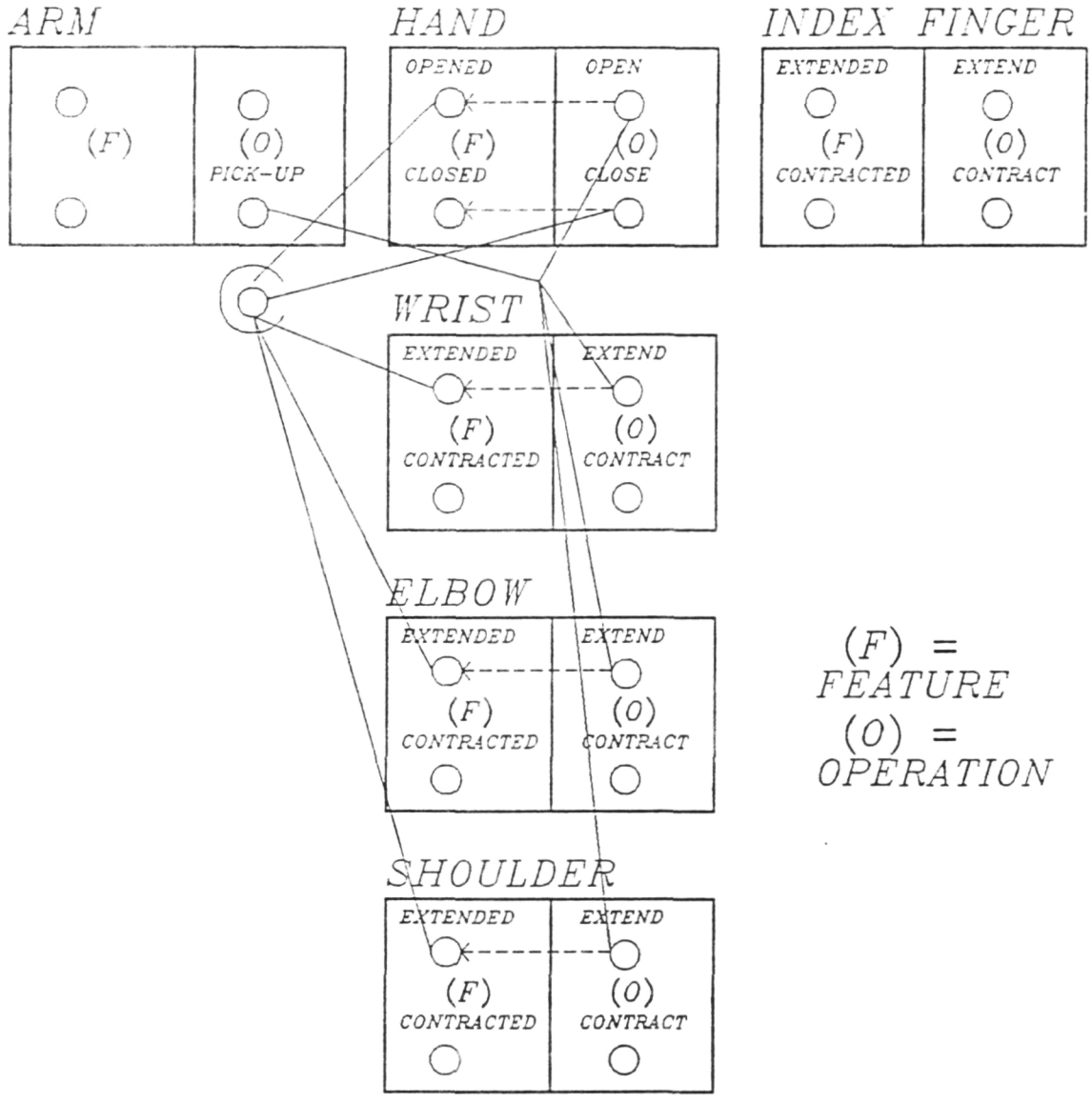
  - Operation hierarchy represents operations (actions) which the net is capable of executing in response to detected features. Operations are decomposed into sequences of sub-operations.

  - Cross-coupling from features to operations provides "rule-like" firing of operations as triggering features become active.

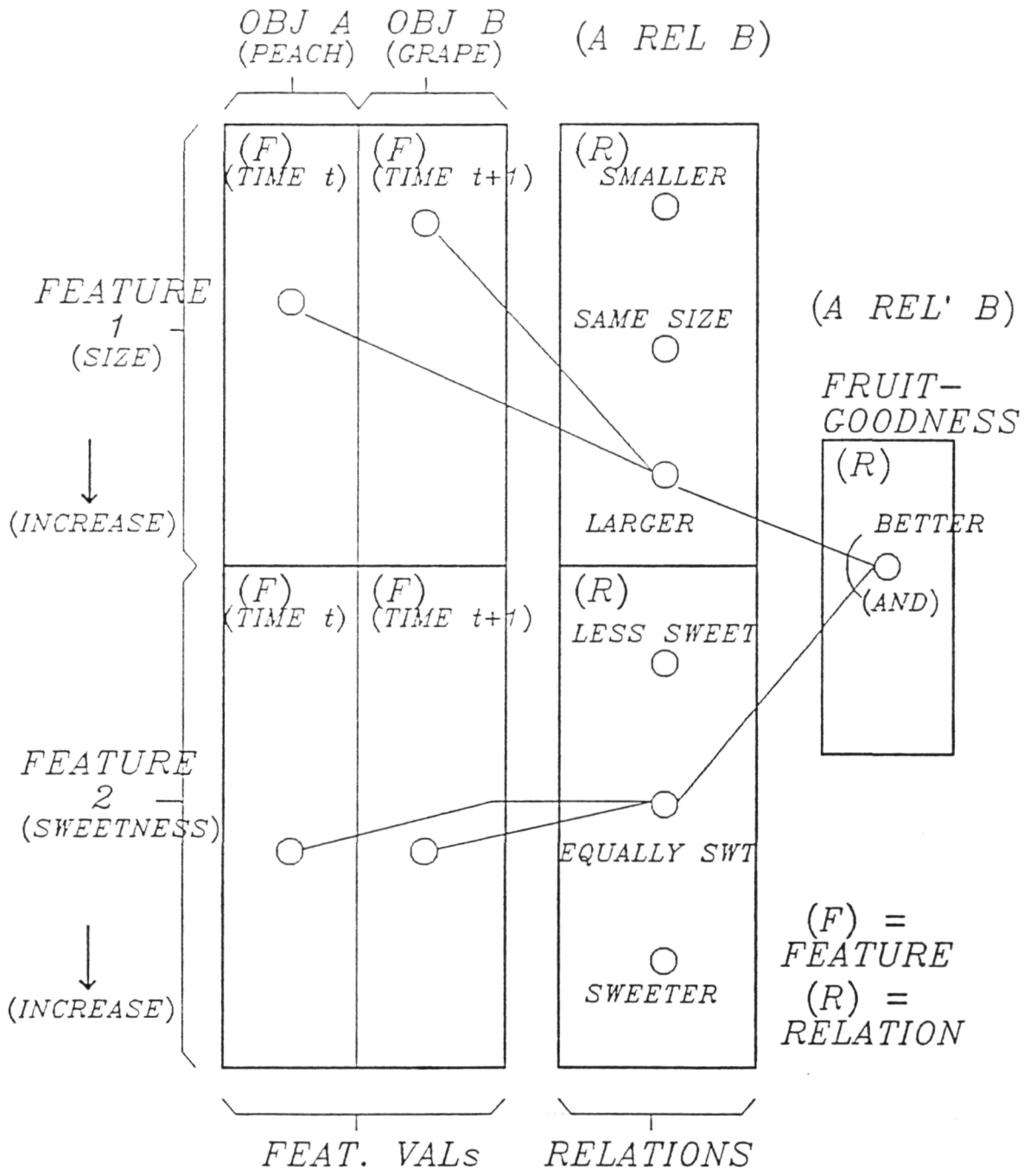
  - Cross-coupling from operations to features activates "expectations" of state-changes which should result from execution of operations. This allows operations to execute in "closed-loop" mode.


ENVIRONMENT
Inputs
Outputs
Feature 0
Operation 0
Feature i
Operation i
Expectations
Biasing
Implication
Context
Sequencing
Decomposition
Feature i+1
Operation i+1
Feature n
Operation n


COLOR (F)
GRN
YEL
RED
AND
PIPPIN (F)
AND
APPLE
GOLD DELIC
OR
AND
RED DELIC
(F)
SWEET
BITTER
FLAVOR
(F)
FRUIT
OR
(F)
CRISP
GRAINY
TEXTURE
(F)
AND
POMEGRANATE
(F) = FEATURE


ARM
(F)
(O)
PICK-UP
HAND
OPENED
OPEN
(F)
CLOSED
(O)
CLOSE
INDEX FINGER
EXTENDED
EXTEND
(F)
CONTRACTED
(O)
CONTRACT
WRIST
EXTENDED
EXTEND
(F)
CONTRACTED
(O)
CONTRACT
ELBOW
EXTENDED
EXTEND
(F)
CONTRACTED
(O)
CONTRACT
SHOULDER
EXTENDED
EXTEND
(F)
CONTRACTED
(O)
CONTRACT
(F) = FEATURE
(O) = OPERATION


PICK-UP (OBJECT)
OPEN HAND
EXTEND WRIST
EXTEND ELBOW
EXTEND SHOULDER
CLOSE HAND


OBJ A (PEACH)
OBJ B (GRAPE)
(A REL B)
(F) (TIME t)
(F) (TIME t+1)
(R) SMALLER
SAME SIZE
LARGER
FEATURE 1 (SIZE)
(INCREASE)
(A REL' B)
FRUIT-GOODNESS
(R)
BETTER
(AND)
(F) (TIME t)
(F) (TIME t+1)
(R) LESS SWEET
FEATURE 2 (SWEETNESS)
EQUALLY SWT
SWEETER
(INCREASE)
(F) = FEATURE
(R) = RELATION
FEAT. VALs
RELATIONS

Why Build KRN's from PAN's ?

- Need to represent *events* (features and operations):
  - Events are discrete and uniquely-identifiable
  - Represent events by distinct PAN nodes
  - Detection of feature has a "certainty" measure (strength of belief); execution of operation has a "priority" measure (degree of appropriateness)
  - Encode in (a pattern of) continuous-valued PAN node activation levels
- Need to represent *relationships between events*:
  - Events favor or inhibit other events with some pair-wise "degree of causality" (implication strength)
  - Represent events by distinct (positive or negative) PAN links

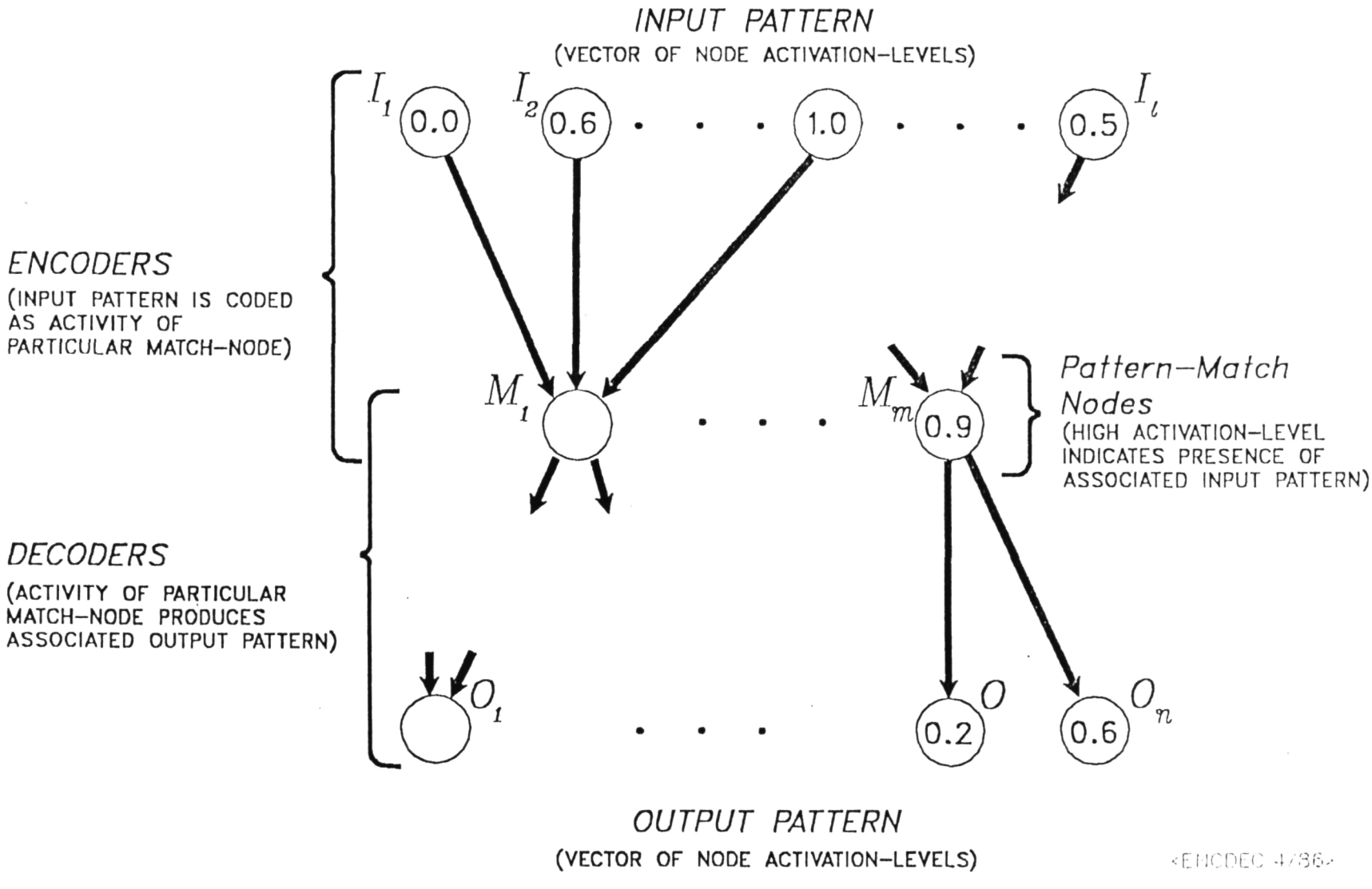
- Adaptiveness *alters beliefs about degree of influence* of one event on another:
  - Modify pair-wise degree of causality
  - Use FAN-link "learning" mechanism (e.g. Hebb model)
- Operations produce *sequences of events*:
  - Arbitration between candidate operations requires comparison of candidates' "appropriateness" measure
  - Encode in FAN node activation levels
  - Temporal constraints important in executing some operations (use network dynamics)
  - Event sequencing requires "handshaking" (use directed FAN flow of activation)
- Basic functions to *implement behavior* (stimulus-response cycle):
  - Detect occurrence of events (like IF-clauses; use FAN "pattern-encoder" circuits)

- Internal events represent responses (like THEN-clauses; use FAN "decoder circuits" to produce activity patterns)

- Events and operations may *occur independently*, and should be processed independently:

  - FAN nodes and links are asynchronous independent processors

  - FAN processing is local (events and operations in FAN nodes, inter-node dependencies through FAN links)


INPUT PATTERN
(VECTOR OF NODE ACTIVATION-LEVELS)
I_1
0.0
I_2
0.6
1.0
0.5
I_i
ENCODERS
(INPUT PATTERN IS CODED AS ACTIVITY OF PARTICULAR MATCH-NODE)
M_1
M_m
0.9
Pattern-Match Nodes
(HIGH ACTIVATION-LEVEL INDICATES PRESENCE OF ASSOCIATED INPUT PATTERN)
DECODERS
(ACTIVITY OF PARTICULAR MATCH-NODE PRODUCES ASSOCIATED OUTPUT PATTERN)
O_1
0.2
O_j
0.6
O_n
OUTPUT PATTERN
(VECTOR OF NODE ACTIVATION-LEVELS)
<ENCDEC 4/86>

Basic KRN "Building-Blocks"

- Features are represented by FAN "encoder" circuits:
  - Output becomes active IF all required input-feature values are active (present)
  - KRN net inputs can drive encoder input features.
  - Match performed by encoder is "fuzzy"— there is some tolerance for input features within a satisfactory range, not just one exact value.
  - Sets of encoders may take input from same set of input features. Inter-encoder competition is used to maximize activity of single encoder with best match to current inputs.
  - FAN "column" circuit acts as encoder; FAN "hypercolumn" circuit acts as competing set of encoders with shared inputs.
- "Decoders" produce a given output activity pattern across a set of output nodes:
  - In producing specific network state-transitions, decoders act like production-rule THEN-clauses.

- ♦ Decoder input activity level scales output activity vector.

- ♦ Decoder outputs can produce output from the overall KRN net.

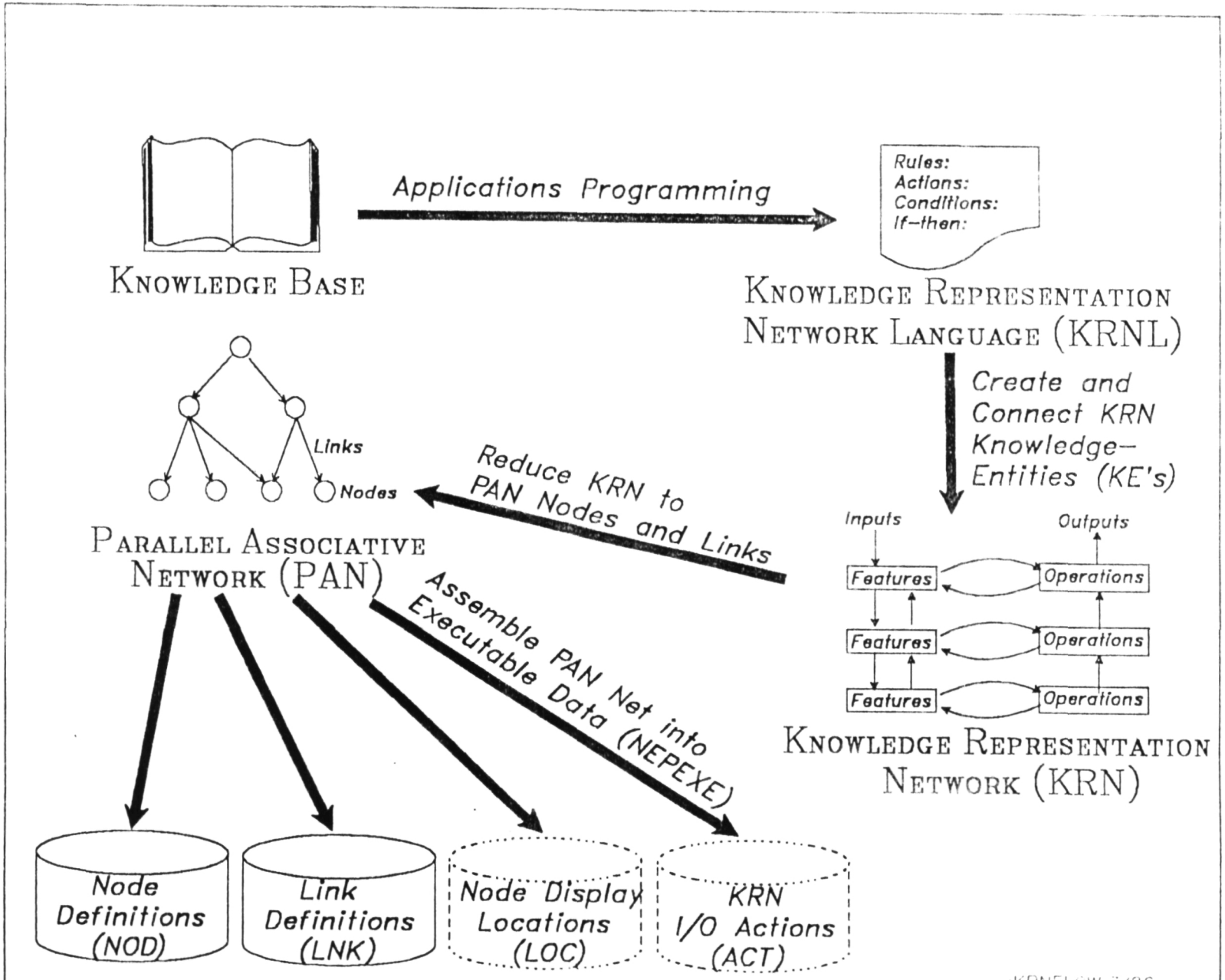
- Encoders cascaded into decoders form "fuzzy" state-machine:

  - ♦ Encoders detect current network state;

  - ♦ Active decoders trigger associated decoders (via FAN links);

  - ♦ Active decoders cause state-transition (i.e. change activity of features which feed encoders).

- Local FAN "attention mechanism" can be used to "enable" or "disable" sets of encoders and decoders. This helps focus processing. (Use and implementation of attention is crucial current research area).


Knowledge Base
Applications Programming
Rules:
Actions:
Conditions:
If–then:
Knowledge Representation
Network Language (KRNL)
Create and
Connect KRN
Knowledge–
Entities (KE's)
Inputs
Outputs
Features
Operations
Features
Operations
Features
Operations
Knowledge Representation
Network (KRN)
Reduce KRN to
PAN Nodes and Links
Links
Nodes
Parallel Associative
Network (PAN)
Assemble PAN Net into
Executable Data (NEPEXE)
Node
Definitions
(NOD)
Link
Definitions
(LNK)
Node Display
Locations
(LOC)
KRN
I/O Actions
(ACT)
<KRNFLOW 5/86>