

1

Intelligent Autonomous Agents
ICS 606 / EE606
Fall 2011

Nancy E. Reed
nreed@hawaii.edu

2

Lecture #8 – Working Together
Outline

- Cooperative distributed problem solving
- Task sharing and result sharing
- Hearsay II example
- Multiagent planning
- References
 - Wooldridge, Ch. 8

3

Working Together

- Why and how do agents work together?
- Important to make a distinction between:
 - *benevolent agents* and
 - *self-interested agents*.

4

Working Together

- Issue: Collaboration, Coordination
- Distinctions between “traditional” Distributed Systems and MAS:
 - Agents are self-interested → interactions resemble games
 - Agents: Decision making and coordination and collaboration dynamically at run-time (vs. at design time in DS)

5

Benevolent Agents

- If we “own” the whole system, we can design agents to **help each other whenever asked**.
- In this case, we can assume agents are **benevolent**: our best interest is their best interest.
- Problem-solving in benevolent systems is **cooperative distributed problem solving (CDPS)**.
- *Benevolence simplifies the system design task enormously!*

6

Self-Interested Agents

- If agents represent individuals or organizations, (the more general case), then we cannot make the benevolence assumption
- Agents will be assumed to **act to further their own interests**, possibly at expense of others
- Potential for **conflict**
- May complicate the design task enormously

7

Task Sharing and Result Sharing

- Two main modes of cooperative problem solving:
 - *task sharing* - components of a task are distributed to component agents
 - *result sharing* - information (partial results etc) is distributed and communicated between agents

8

Result Sharing Issues

- Confidence
 - Independently derived solutions can be cross-checked
- Completeness
 - Agents share local views to achieve better global view
- Precision
 - Agents share results to ensure that overall solution precision is increased
- Timeliness
 - Result Sharing produces more in faster problem-solving

9

Solution Design/Synthesis

- Decomposition
 - Hierarchical
 - Non-hierarchical
- Sub-problems
 - Independent – easier to combine to solve entire problem
 - Dependent/interactions – solution to parts must be combined or modified to solve entire problem

10

Cooperative Distributed Problem Solving (CDPS)

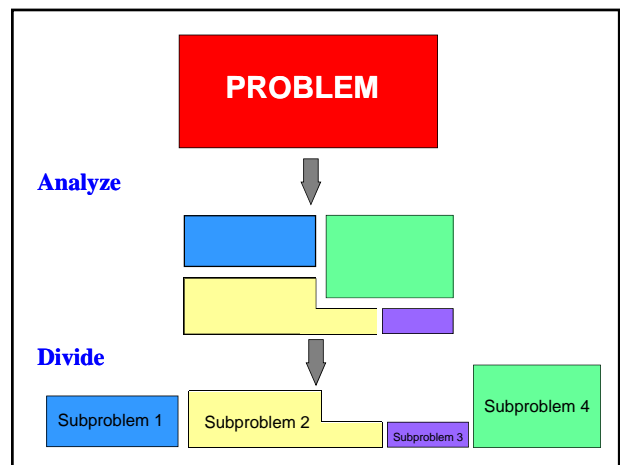
- No global control
- No global data storage
- Data and control are distributed

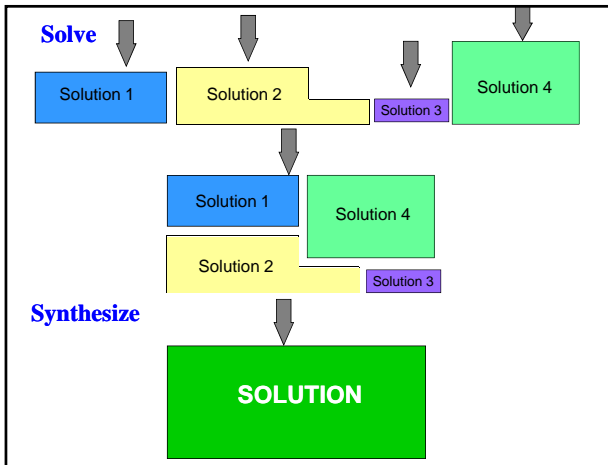
11

Four Solution Phases

1. Problem Decomposition
2. Sub-problem distribution
3. Sub-problem solution
4. Answer synthesis

The contract net protocol deals with phase 2.





14

Distinction between CDPS and Parallel Problem Solving (PPS)

- In PPS nodes are usually **uniform processors**
- In PPS we have **single instance** responsible for **problem decomposition** and **result synthesis**
- Criteria to consider
 - Coherence
 - Coordination

15

CDPS Characteristics and Consequences

- Communication is slower than computation
 - loose coupling
 - efficient protocol
 - modular problems
 - problems with large grain size

16

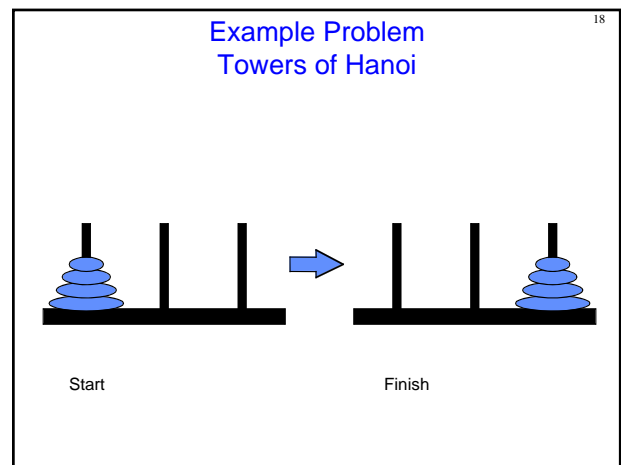
CDPS Characteristics and Consequences (2)

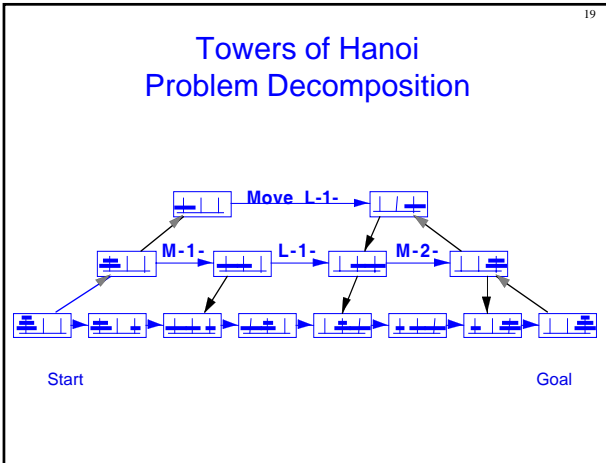
- Any unique node is a **potential bottleneck**
 - distribute data
 - distribute control
 - organized behavior is hard to guarantee (since no one node has complete picture)

17

Problem Decomposition

- **Decomposition granularity**
- ACTOR system decomposed problem until sub-problems are at the level of programming language commands → too fine grained.
- problems with synthesis, management overhead etc. outweigh decomposition advantages
- **Single decomposition** instance vs. **collaborative decomposition**
- **Competence** of agents must be known





20

The Contract Net

- An approach to *distributed problem solving*, focusing on task distribution
- Task distribution viewed as a kind of contract negotiation
- “Protocol” specifies *content* of communication, not just form
- Two-way transfer of information is natural extension of transfer of control mechanisms

21

Contract Net

- The collection of nodes is the “*contract net*”
- Each node on the network can, at different times or for different tasks, be a manager or a contractor
- When a node gets a composite task (or for any reason can’t solve its present task), it breaks it into subtasks (if possible) and announces them (acting as a manager), receives bids from potential contractors, then awards the job (example domain: network resource management, printers, ...)

22

The Contract Net

- Well known **task-sharing** protocol for task allocation is called *contract net*,
- 5 phases
 1. Recognition
 2. Announcement
 3. Bidding
 4. Awarding
 5. Expediting

23

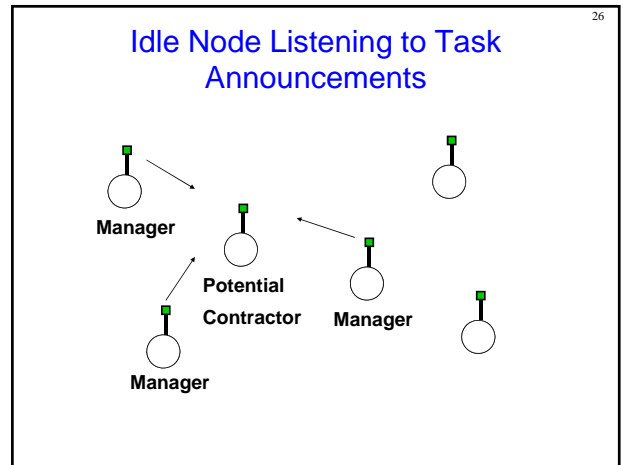
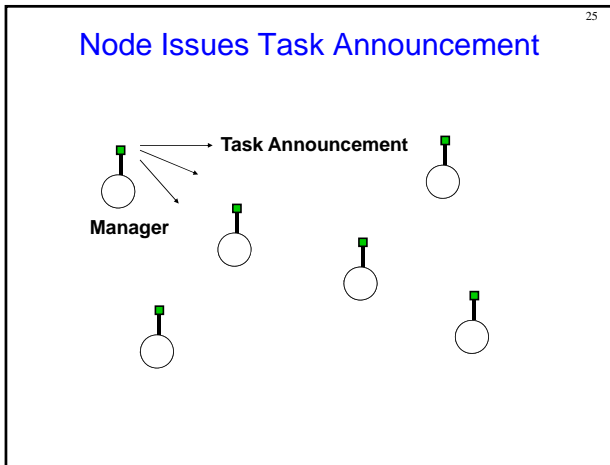
Step 1 - Recognition

- In this stage, an agent recognizes it has a problem it wants help to achieve
- Agent has a goal, and either. . .
 - realizes it **cannot achieve the goal** in isolation — does not have capability;
 - realises it **would prefer not to achieve the goal** in isolation (typically because of solution quality, deadline, etc)

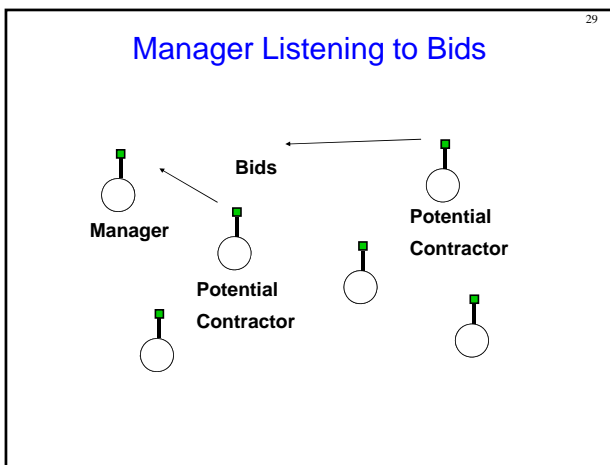
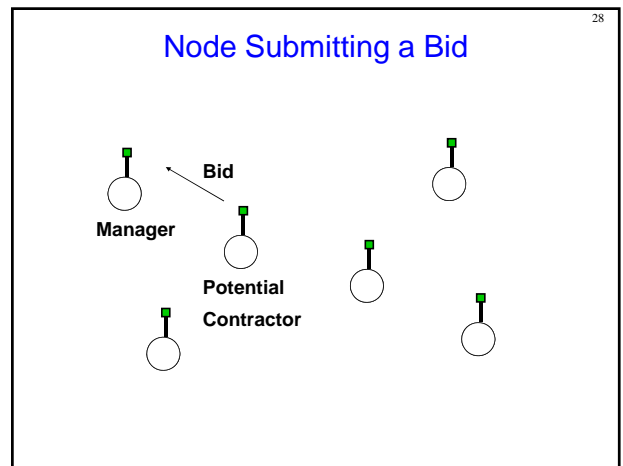
24

Step 2 - Announcement

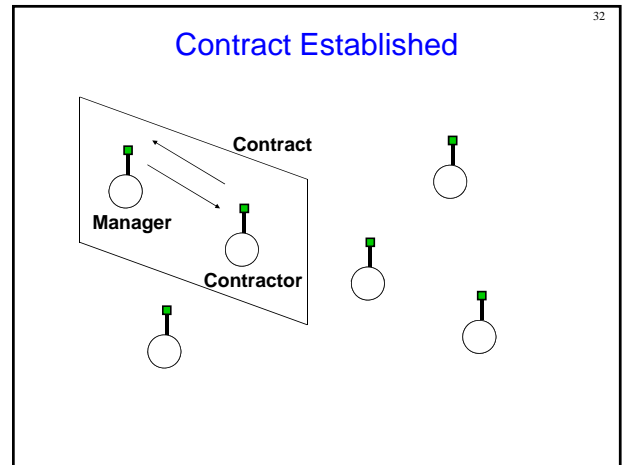
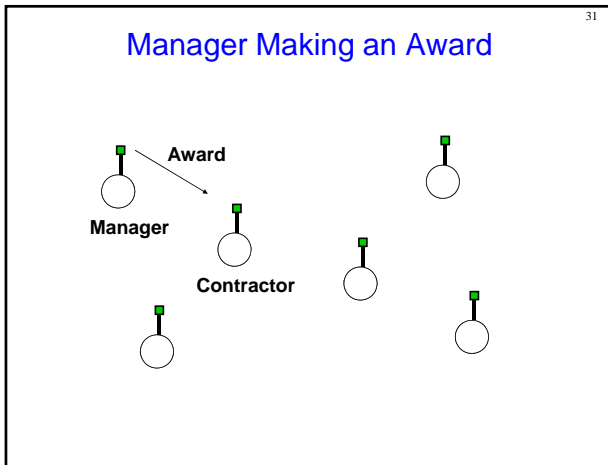
- The agent sends out an announcement which includes a *specification* of the task to be achieved
- Specification must encode:
 - description of task itself (may be executable);
 - any constraints (e.g., deadlines, quality constraints)
 - meta-task information (e.g., “bids must be submitted by_”)
- The announcement is then *broadcast*



- 27
- ### Step 3 - Bidding
- Agents that receive the announcement decide for themselves whether they wish to **bid** for the task
 - Factors:
 - agent must decide whether it is **capable** of expediting task;
 - agent must determine **quality** constraints & **price** information (if relevant)
 - If they do choose to bid, then they submit a **tender**



- 30
- ### Awarding (4) and Expediting (5)
- Agent that sent task announcement must **choose** between bids & decide who to “award the contract” to
 - The result of this process is communicated to agents that submitted a bid
 - The successful **contractor** then expedites the task. May involve generating further manager-contractor relationships: **sub-contracting**



- Issues for Implementing Contract Net**
- How to . . .
 - specify *tasks*?
 - specify *quality of service*?
 - *select* between competing offers?
 - *differentiate* between offers based on multiple criteria?

- Domain-Specific Evaluation**
- Task announcement message prompts potential contractors to use domain specific task evaluation procedures; there is deliberation going on, not just selection — perhaps no tasks are suitable at present
 - Manager considers submitted bids using domain specific bid evaluation procedure

- Types of Messages**
- Task announcement
 - Bid
 - Award
 - Interim report (on progress)
 - Final report (including result description)
 - Termination message (if manager wants to terminate contract)

- Efficiency Modifications**
- Focused addressing — when general broadcast isn't required
 - Directed contracts — when manager already knows which node is appropriate
 - Request-response mechanism — for simple transfer of information without overhead of contracting
 - Node-available message — reverses initiative of negotiation process

Message Format

- Task Announcement Slots:
 - Eligibility specification
 - Task abstraction
 - Bid specification
 - Expiration time

Task Announcement Example (common internode language)

```

To:      *
From:    25
Type:    Task Announcement
Contract: 43-6
Eligibility Specification: Must-Have FFTBOX
Task Abstraction:
    Task Type Fourier Transform
    Number-Points 1024
    Node Name 25
    Position LAT 64N LONG 10W
Bid Specification: Completion-Time
Expiration Time: 29 1645Z NOV 1980
  
```

- The existence of a common internode language allows new nodes to be added to the system modularly, without the need for explicit linking to others in the network
- (e.g., as needed in standard procedure calling)
- or object awareness (as in OOP)

Task Sharing and Result Sharing

- Three stages of CDPS (also applicable to MAS)
 - Problem decomposition
 - Sub-problem solution
 - Solution synthesis
- Problem decomposition
 - Iteratively hierarchically decompose overall problem into smaller sub-problems until agent can solve them
 - Different decomposition levels $\leftarrow \rightarrow$ different levels of abstraction

The Hearsay II Speech Understanding System

- Developed at Carnegie-Mellon in the mid-1970's
- Goal was to reliably interpret connected speech involving a large vocabulary
- First example of the blackboard architecture, "a problem-solving organization that can effectively exploit a multi-processor system." (Fennel and Lesser, 1976)

The Motivations

- Real-time speech understanding required more processor power than could be expected of typical machines in 1975 (between 10 and 100 MIPS); parallelism offered a way of achieving that power
- There are always problems beyond the reach of current computer power—parallelism offers us hope of solving them now
- The complicated structure of the problem (i.e., speech understanding) motivated the search for new ways of organizing problem solving knowledge in computer programs

Result Sharing in Blackboard Systems ⁴³

- The first scheme for cooperative problem solving: the *blackboard system*
- Results shared via shared data structure (BB).
- Knowledge sources (KS)
- Multiple agents (KSs/KAs) can read and write to BB
- Agents write partial solutions to BB
- BB may be structured hierarchically
- Mutual exclusion over BB required bottleneck
- No concurrent activity

Result Sharing in Subscribe/Notify ⁴⁴

- Common design pattern in OO systems: *subscribe/notify*
- An object *subscribes* to another object, saying “tell me when event *e* happens”
- When event *e* happens, original object is notified
- Information pro-actively *shared* between objects
- Objects required to know about the *interests* of other objects → inform objects when relevant information arises

The Blackboard Architecture ⁴⁵

1. Multiple, diverse, independent and asynchronously executing knowledge sources (KS's)
2. Cooperating (in terms of *control*) via a generalized form of hypothesize-and-test, involving the data-directed invocation of KS processes
3. Communicating (in terms of *data*) via a shared blackboard-like database

A “Knowledge Source” (KS) ⁴⁶

“An agent that embodies the knowledge of a particular aspect of a problem domain,” and furthers the solution of a problem from that domain by taking actions based on its knowledge.

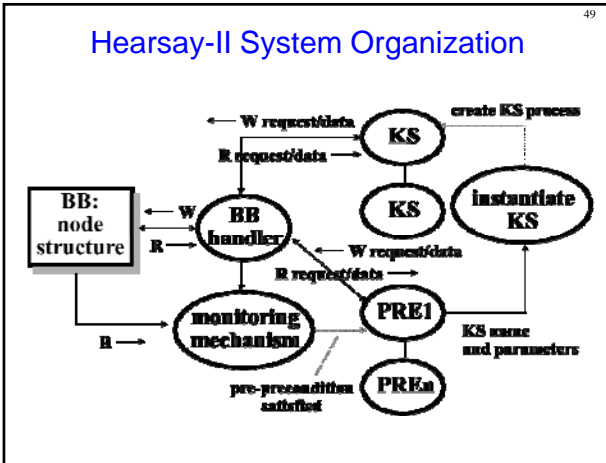
In speech understanding, there could be distinct KS's to deal with acoustic, phonetic, lexical, syntactic, and semantic information.

Abstract Model ⁴⁷

- The blackboard architecture is a parallel production system (productions: $P \rightarrow A$)
- Preconditions are satisfied by current state of the (dynamic) blackboard data structure, and trigger their associated Action
- Actions presumably alter the blackboard data structure
- Process halts when no satisfied precondition is found, or when a “stop” operation is executed (failure or solution)

The Blackboard ⁴⁸

- Centralized multi-dimensional data structure
- Fundamental data element is called a node (nodes contain data fields)
- Readable and writable by any precondition or KS (production action)
- Preconditions are procedurally oriented and may specify arbitrarily complex tests



- Preconditions have “pre-preconditions” that sense primitive conditions on the blackboard, and schedule the real (possibly complex) precondition test
- KS processes are also procedurally oriented, generally hypothesize new data (added to data base) or verify or modify data already in the data base

- Hypothesize-and-test paradigm — hypotheses representing partial problem solutions are generated and then tested for validity
- Neither precondition procedures nor action procedures are assumed to be “indivisible”; activity is occurring concurrently (multiple KS’s, multiple precondition tests...)

- Multi-dimensional Blackboard**
- For example, in Hearsay-II, the system data base had three dimensions for nodes:
 - informational level (e.g., phonetic, surface-phonemic, syllabic, lexical, and phrasal levels)
 - utterance time (speech time measured from beginning of input)
 - data alternatives (multiple nodes can exist simultaneously at the same informational level and utterance time)

- Modularity**
- The “KS’s are assumed to be independently developed” and don’t know about the explicit existence of other KS’s — communication must be indirect
 - Motivation: the KS’s have been developed by many people working in parallel; it is also useful to check how the system performs using different subsets of KS’s

- KS Communication**
- Takes two forms:
 - Database monitoring to **collect data event information** for future use (local contexts and precondition activation)
 - Database monitoring to **detect data events that violate** prior data assumptions (tags and messages)

55

Local Contexts

- Each precondition and KS process that needs to remember the history of database changes has its own local database (local context) that keeps track of the global database changes that are relevant to that process
- When a change (data event) occurs on the blackboard, the change is broadcast to all interested local contexts (data node name and field name, with old value of field)
- The blackboard holds only the most current information; local contexts hold the history of changes

56

Data Integrity

- Because of the concurrency in blackboard access by preconditions and KS's (and the fact that they are not indivisible), there is a need to maintain data integrity:
 - **Syntactic** (system) integrity: e.g., each element in a list must point to another valid list element
 - **Semantic** (user) integrity: e.g., values associated with adjacent list elements must be always less than 100 apart

57

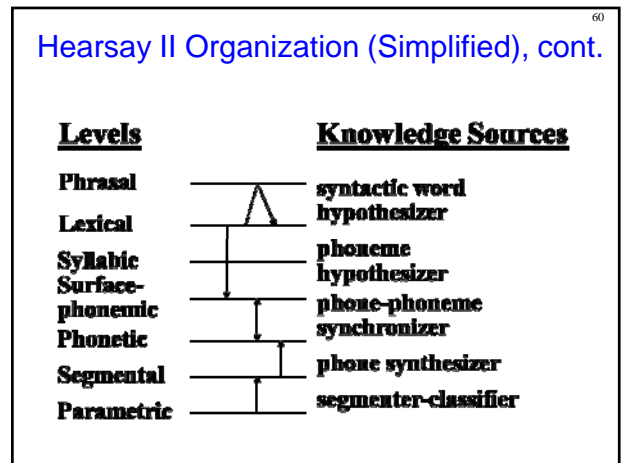
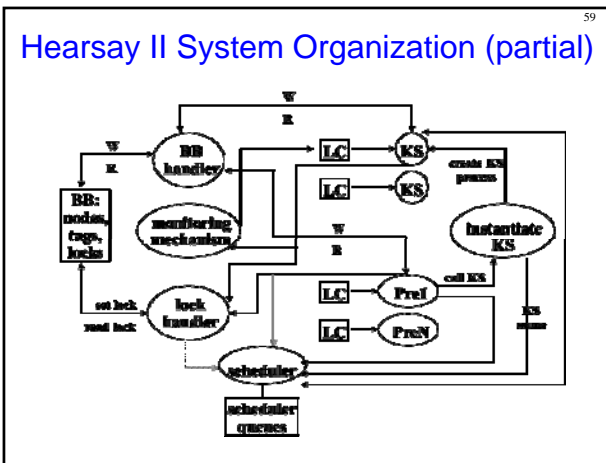
Locks

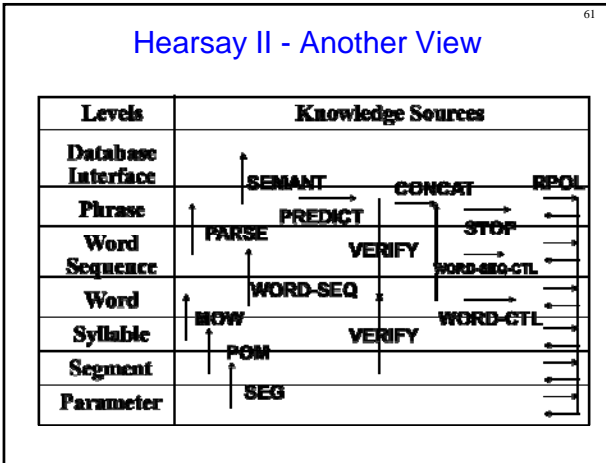
- Locks allow several ways for a process to acquire exclusive or read-only data access:
 - Node locking (specific node)
 - Region locking (a collection of nodes specified by their characteristics, e.g., information level and time period)
 - Node examining (read-only access to other processes)
 - Region examining (read-only)
 - Super lock (arbitrary group of nodes and regions can be locked)

58

Tagging

- Locking can obviously cut down on system parallelism, so the blackboard architecture allows data-tagging:
 - Data assumptions placed into the database (defining a critical data set); other processes are free to continue reading and writing that area, but if the assumptions are invalidated, warning messages are sent to relevant processes
 - Precondition data can be tagged by the precondition process on behalf of its KS, so that the KS will know if the precondition data has changed before action is taken





The KS's

Signal Acquisition, Parameter Extraction, Segmentation and Labeling:
 SEG: Digitizes the signal, measures parameters, produces labeled segmentation

Word Spotting:
 POM: Creates syllable-class hypotheses from segments
 MOW: Creates word hypotheses from syllable classes
 WORD-CTL: Controls the number of word hypotheses that MOW creates

The KS's

Phrase-Island Generation:
 WORD-SEQ: Creates word-sequence hypotheses that represent potential phrases, from word hypotheses and weak grammatical knowledge
 WORD-SEQ-CTL: Control the number of hypotheses that WORD-SEQ creates
 PARSE: Attempts to parse a word-sequence and, if successful, creates a phrase hypothesis from it

The KS's

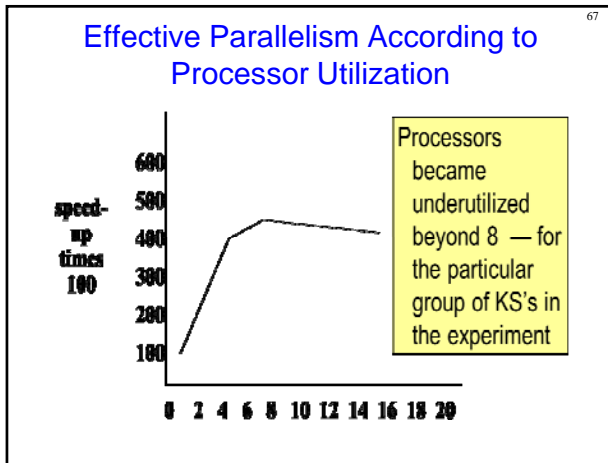
Phrase Extending:
 PREDICT: Predicts all possible words that might syntactically precede or follow a given phrase
 VERIFY: Rates the consistency between segment hypotheses and a contiguous word-phrase pair
 CONCAT: Creates a phrase hypothesis from a verified, contiguous word-phrase pair

The KS's

Rating, Halting, and Interpretation:
 RPOL: Rates the credibility of each new or modified hypothesis, using information placed on the hypothesis by other KS's
 STOP: Decides to halt processing (detects a complete sentence with a sufficiently high rating, or notes the system has exhausted its available resources), and selects the best phrase hypothesis (or a set of complementary phrase hypotheses) as the output
 SEMANT: Generates an unambiguous interpretation for the information-retrieval system which the user has queried

Timing statistics (non-overlapping)

- Blackboard reading 16%
- Blackboard writing 4%
- Internal computations of processes 34%
 - Local context maintenance 10%
 - Blackboard access synchronization 27%
 - Process handling 9%
 - (i.e., multiprocess overhead almost 50%)



- Now We Want Distributed Interpretation... 68
- Sensor networks (low-power radar, acoustic, or optical detectors, seismometers, hydrophones...)
 - Network traffic control
 - Inventory control
 - Power network grids
 - Mobile robots

- Distributed Interpretation 69
- Working Assumption Number 1: Interpretation techniques that search for a solution by the incremental aggregation of partial solutions are especially well-suited to distribution
 - Errors and uncertainty from input data and incomplete or incorrect knowledge are handled as an integral part of the interpretation process
 - Working Assumption Number 2: Knowledge-based AI systems can handle the additional uncertainty introduced by a distributed decomposition without extensive modification

- Distributed Interpretation 70
- The early experiments with distributing Hearsay-II across processors were simple; later experiments (e.g., the DVMT) were much more rigorous:
 1. At first, few (only 3) nodes
 2. Few experiments (heavy simulation load)
 3. "There is probably no practical need for distributing a single-speaker speech-understanding system."

- How Do We Go About Distributing? 71
- Options:
 - Distribute information (the blackboard is multi-dimensional — each KS accesses only a small subspace)
 - Distribute processing (KS modules are largely independent, anonymous, asynchronous)
 - Distribute control (send hypotheses among independent nodes, activating KS's)

- Distributed Interpretation 72
- The multi-processor implementation of Hearsay-II, with explicit synchronization techniques to maintain data integrity, achieved a speed-up factor of six — but the need for any synchronization techniques is a bad idea for a true distributed interpretation architecture

The Uni-Processor and Synchronized Multi-processor Versions...

73

1. The scheduler (which requires a global view of pending KS instantiations [scheduling queues] and the focus-of-control database) is centralized
2. The blackboard monitor (updating focus-of-control database and scheduling queues) is centralized
3. Patterns of KS blackboard access overlap, hard to have compartmentalized subspaces

Distributed Interpretation

74

- In fact, the explicit synchronization techniques could be eliminated, and the speedup factor increased from 6 to 15
- All sorts of internal errors occurred because of the lack of centralized synchronization, but the architecture was robust enough to (eventually) correct these errors

Dimensions of Distribution - Information

75

- Distribution of the blackboard:
 - Blackboard is distributed with no duplication of information
 - Blackboard is distributed with possible duplication, synchronization insures consistency
 - Blackboard is distributed with possible duplications and inconsistencies

Information, cont.

76

- Transmission of hypotheses:
 - Hypotheses are not transmitted beyond the local node that generates them
 - Hypotheses may be transmitted directly to a subset of nodes
 - Hypotheses may be transmitted directly to all nodes

Dimensions of Distribution - Processing

77

- Distribution of KS's:
 - Each node has only one KS
 - Each node has a subset of KS's
 - Each node has all KS's
- Access to blackboard by KS's:
 - A KS can access only the local blackboard
 - A KS can access a subset of nodes' blackboards
 - A KS can access any blackboard in the network

Dimensions of Distribution - Control:

78

- Distribution of KS activation:
 - Hypothesis change activates only local node's KS's
 - Change activates subset of nodes' KS's
 - Change activates KS's in any node
- Distribution of scheduling/focus-of-control:
 - Each node does its own scheduling, using local information
 - Each subset of nodes has a scheduler
 - A single, distributed database is used for scheduling

79

Two Ways of Viewing the Distribution of Dynamic Information

1. There is a virtual global database; local nodes have partial, perhaps inconsistent views of the global database
2. Each node has its own database; the union of these across all nodes, with any inconsistencies, represents the total system interpretation — not a system that's been distributed, but a network of cooperating systems

80

Focusing the Nodes

- The blackboard is multi-dimensional: one dimension might be the information level
- Other dimensions, orthogonal to the information level, fix the location of the event which the hypothesis describes:
 - *signal interpretation*: physical location
 - *speech understanding*: time
 - *image understanding*: 2 or 3 dimensional space
 - *radar tracking*: 3 dimensional space

81

- All levels of the system, together with the full extent of the location dimension(s), define the largest possible scope of a node
- The area of interest of a node is the portion of this maximum scope representable in the node's local blackboard
- The location segment extends beyond the range of the local sensor (to allow the node to acquire context information from other nodes)
- At higher levels, the location dimension tends to get larger

82

Coherence and Coordination

- Coherence: Refers to “how well the MAS behaves as a unit along some dimension of evaluation” (Bond et al.)
- Coherence may be measured in terms of
 - solution quality
 - resource usage
 - conceptual clarity of operation
 - performance degradation if unexpected failure occurs

83

Example of Areas of Interest

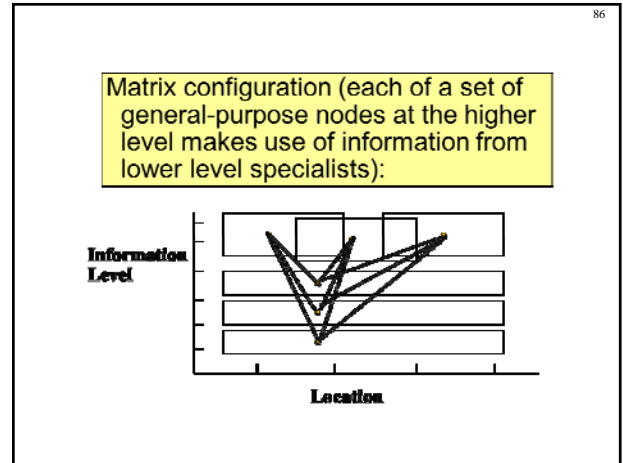
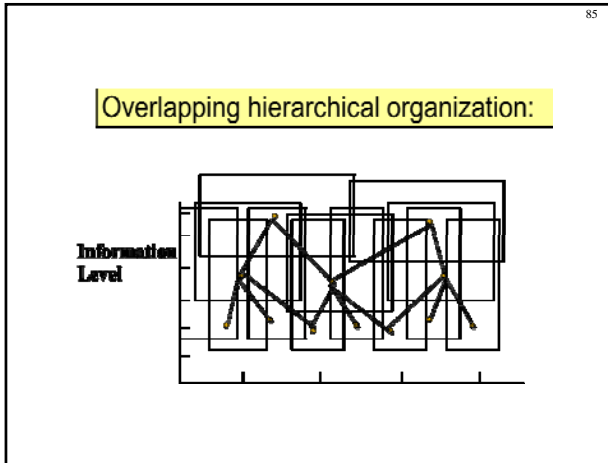
The diagram illustrates the expansion of an area of interest across three levels. The horizontal axis represents location from 0 to 100. Level 1 shows a narrow vertical bar at location 50. Level 2 shows a wider vertical bar centered at 50. Level 3 shows a very wide vertical bar spanning from approximately 25 to 75. Two Knowledge Sources, KS1 and KS2, are indicated by arrows pointing to specific locations within the Level 1 and Level 2 bars.

84

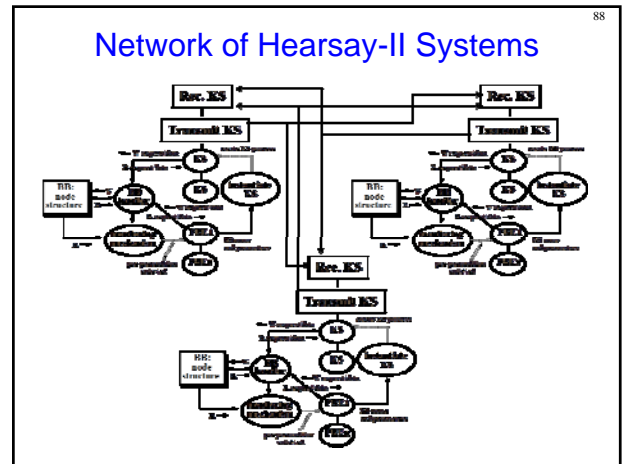
Network Configurations

All nodes contain the same set of KS's and levels — the configuration is flat:

The diagram shows a flat network configuration. The vertical axis is labeled 'Information Level' and the horizontal axis is labeled 'Location'. Three vertical bars represent nodes at different locations. Each node has a horizontal line across its width, indicating that all nodes share the same set of knowledge sources and levels, resulting in a flat configuration.



- 87
- Internode Communication**
- In Hearsay-II, all inter-KS communication is handled by the creation, modification, and inspection of hypotheses on the blackboard
 - In the distributed Hearsay-II architecture, inter-node communication is handled the same way
 - Added to the local node's KS's is a RECEIVE KS and a TRANSMIT KS



- 89
- Internode Communication**
- In general, communication occurs to "nearby" nodes, based on the location dimensions and overlapping areas of interest
 - As a heuristic this makes sense: close nodes are likely to be most interested in your information (and have interesting information for you)
 - Those are also the nodes with whom it is cheapest to communicate

- 90
- Why Not Continue Using the Hearsay-II Domain?**
- Time-consuming to run the simulation, since the underlying system was large and slow
 - The speech task didn't naturally extend to larger numbers of nodes (partly because the speech understanding problem has one-dimensional [time] sensory data)

91

- Hearsay-II had been tuned, for efficiency reasons, so that there was a “tight-coupling among knowledge sources and the elimination of data-directed control at lower blackboard levels” — in direct contradiction of the overall system philosophy! Tight coupling causes problems with experimentation (e.g., eliminating certain KS’s)
- The KS code was large and complex, so difficult to modify

92

Future Directions

- Other domains
- Test beds for experimentation

93

Summary

- Distributed Problem-Solving and Planning
- Hearsay II example

94

Questions

