Chapter 31. Intelligent System Architectures

The Quest for Artificial Intelligence, Nilsson, N. J., 2009.

Lecture Notes on Artificial Intelligence, Spring 2012

Summarized by Jang, Ha-Young and Lee, Chung-Yeon

Biointelligence Laboratory
School of Computer Science and Engineering
Seoul National Univertisy

http://bi.snu.ac.kr

Contents

Overview of Chapter 31

- Computer scientists have developed various ways to put together large programs consisting of many specialist subprograms. This scheme is the so-called von Neumann architecture.
- The earliest AI programs ran on computers that used the von Neumann architecture, and thus it was natural for the architecture of the programs (that is, the way they themselves were organized) to adhere to the von Neumann style of the computer's operation.
- In contrast with the von Neumann idea of executing instructions one after another in sequence, one can conceive of an architecture in which many instructions are executed simultaneously. For example, in the nonsymbolic world of neural networks, one could imagine groups of neural elements operating simultaneously, even though simulations of these networks have to consider each neural element in turn sequentially.
- In this chapter, I'm going to describe some of the ways researchers have organized their programs to achieve intelligent behavior. Some of them were inspired mainly by engineering and computational considerations and some by cognitive science in its attempt to model psychological data. Some were even inuenced by ideas about how various brain regions function. Parallel operation is assumed in many of these architectures, even though it is often of the simulated variety.

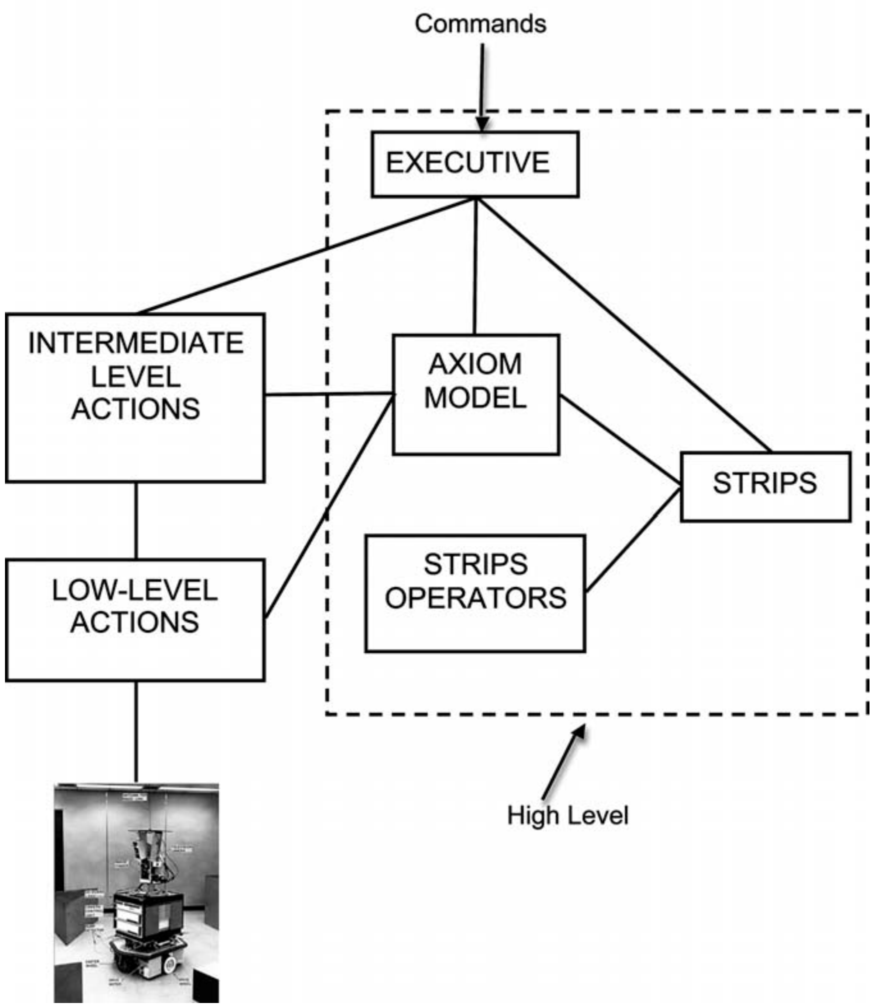
Chapter 31. Intelligent System Architectures

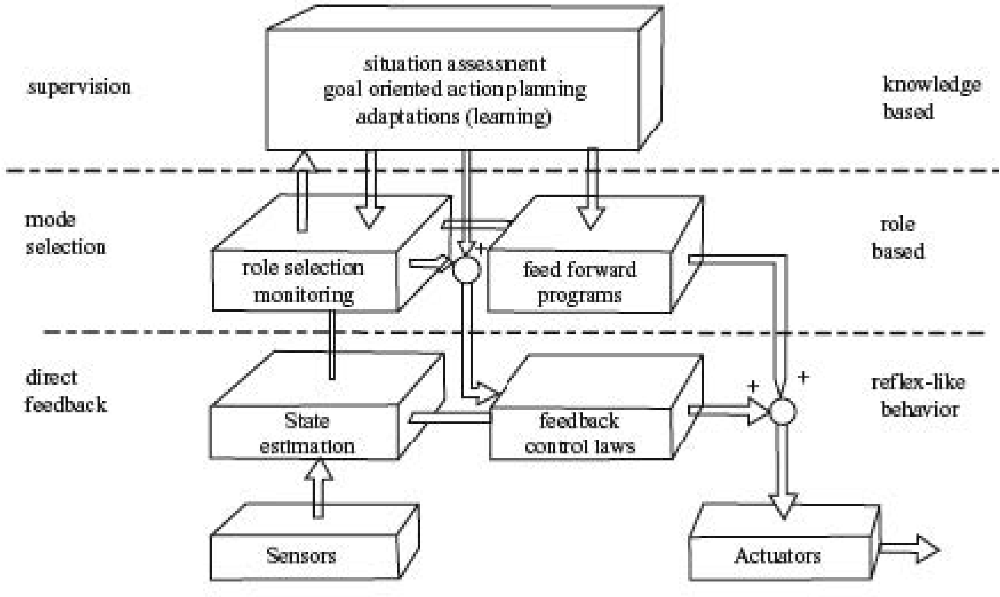
31.1 Computational Architectures

31.1.1 Three-Layer Architectures

- The three-layer architecture arises from the empirical observation that efective algorithms for controlling mobile robots tend to fall into three distinct categories
  - Reactive control algorithms which map sensors directly onto actuators with little or no internal state
  - Algorithms for governing routine sequences of activity which rely extensively on internal state but perform no search
  - Time-consuming (relative to the rate of change of the environment) search-based algorithms such as planners.

31.1.1 Three-Layer Architectures



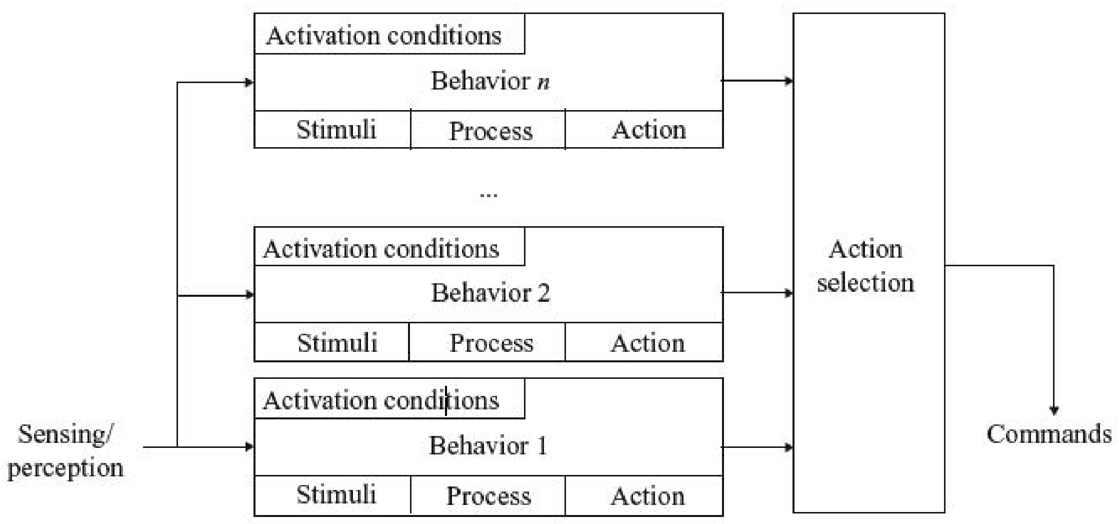


A three-layered architecture for a driverless automobile.

Shakey's three-level architecture.

31.1.2 Multilayered Architectures

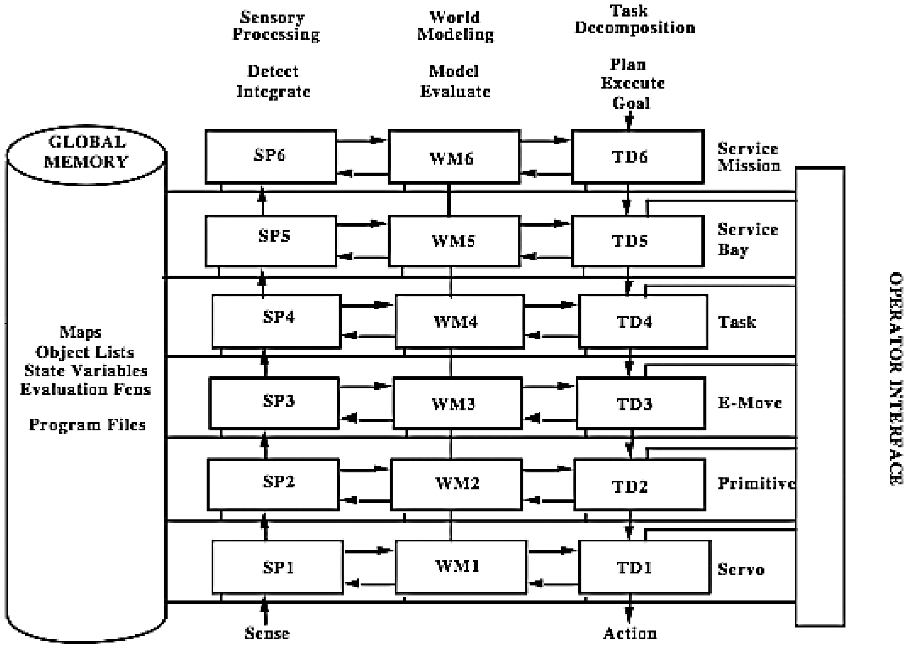
- Behavior-based architecture
  - Composed of specifically programmed robot behaviors
  - Alternative to the three-layered schemes, all of which involved a planning level



A behavior-based architecture

31.1.2 Multilayered Architectures

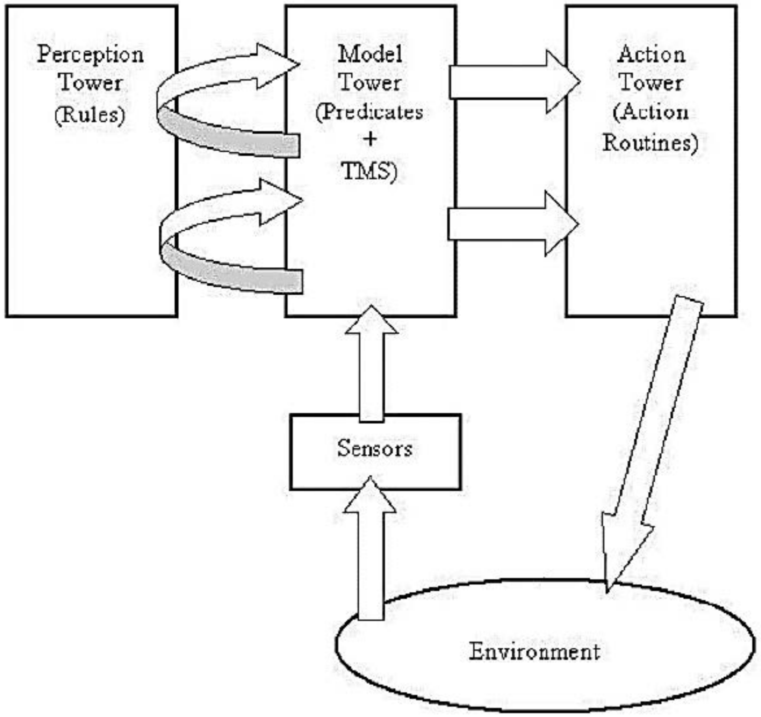
- Reference model architecture
  - Developed by James Albus
  - Multiple layers of real-time control systems(RCSs)
  - A layered structure of RCSs, called NASREM was proposed as the architecture for a flight tele-robotic servicer on the space station.



The NASREM architecture

31.1.2 Multilayered Architectures

- Triple-tower architecture
  - Perception tower: rules
  - Model tower: predicates + truth maintenance system
  - Action tower: action routines

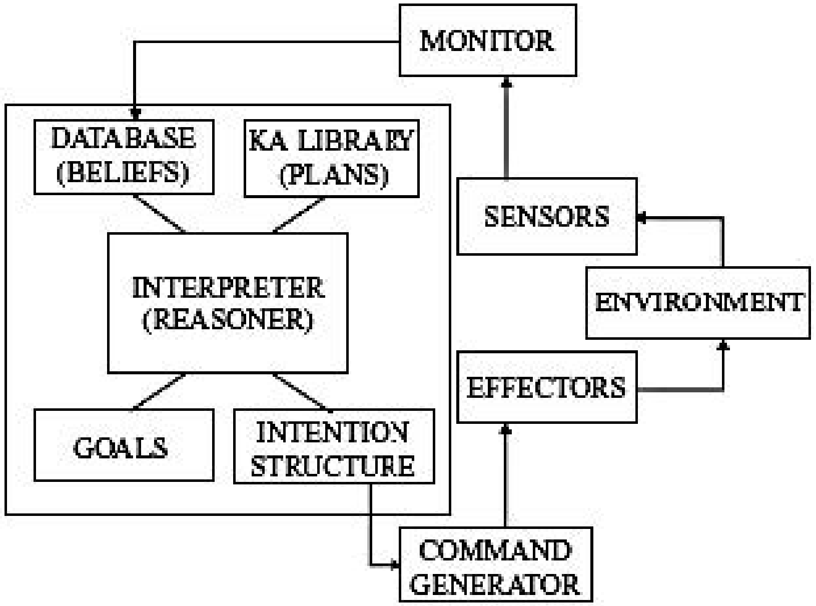

Perception Tower (Rules)
Model Tower (Predicates + TMS)
Action Tower (Action Routines)
Sensors
Environment

Triple Tower Architecture

31.1.3 The BDI Architecture

- Agent architectures based on the philosophical concepts of beliefs, desires, and intentions
  - The database consists of the agent's current beliefs about its environment
  - Goals (the agent's desires) are conditions to be achieved and can refer
  - The KA library of plans contains what are called “Knowledge Areas”
  - The intention structure contains tasks that the system has chosen to execute.
  - The interpreter runs the system.

Michael Georgeff



PRS, a BDI architecture

31.1.4 Architectures for Groups of Agents

- Early work in multiagent systems
  - DISTRIBUTED HEARSAY-II by Victor Lesser and Lee Erman
  - Distributed Vehicle Monitoring Testbed focused on tracking vehicle motion using a distributed sensor network and was a resource for testing methods of cooperative distributed problem solving.
- Multi-Agent Computing Environment (MACE)
  - Developed by Les Gasser
- RoboCup
  - An international joint project to foster AI and intelligent robotics research by providing a standard problem

Manuela Veloso (left) and soccer-playing Aibo robots (right)

31.1.4 Architectures for Groups of Agents

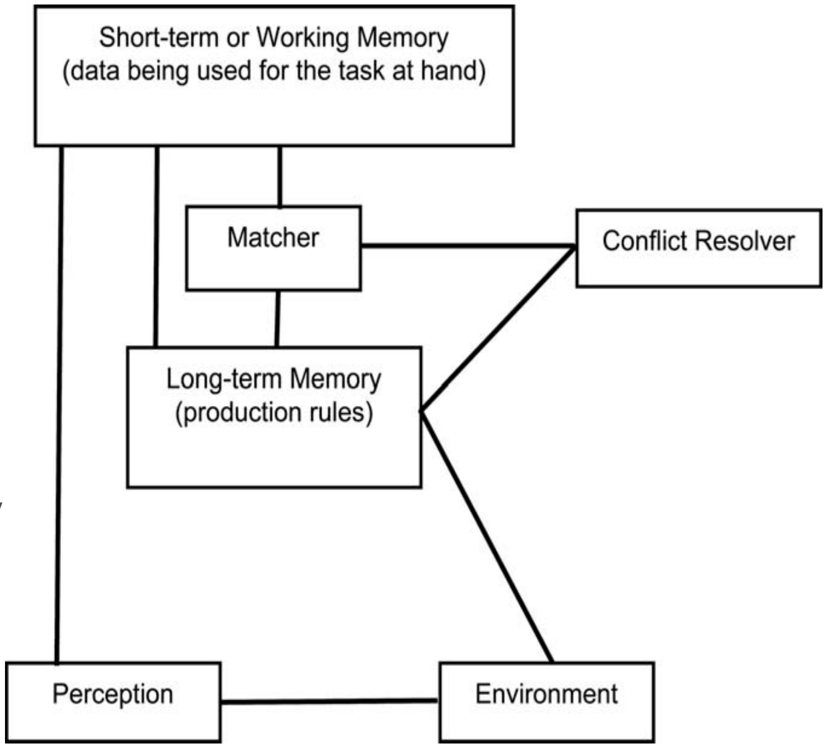
- KQML uses KIF (Knowledge Interchange Format), a language based on first-order predicate calculus, for expressing the content of a message. So, when agent A wants to send a message to agent B, it encodes the content of the message in KIF and then wraps it in the appropriate KQML communicative action. For example, here is a typical KQML/KIF dialog:
- $A\ to\ B: \left(ask - if\ \left(> (size\ chip1)(size\ chip2)\right)\right)$
- $B\ to\ A: (reply\ true)$
- $B\ to\ A: \left(inform\ (= (size\ chip1)\ 20)\right)$
- $B\ to\ A: (inform\ (= (size\ chip2)\ 18))$

Chapter 31. Intelligent System Architectures

31.2 Cognitive Architectures

31.2.1 Production Systems

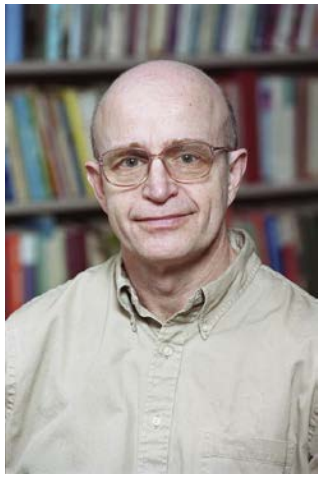
- Models that used IF-THEN rules, called productions by Allen Newell and Herb Simon
  - IF part was a condition
  - THEN part was an action
  - Long-term memory
    - The production rules
  - Short-term or working memory
    - The dynamic information about the task being worked on

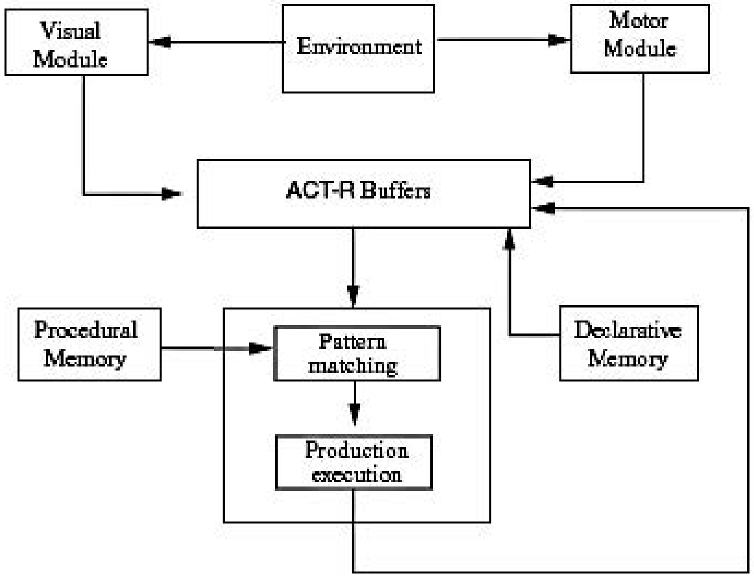

Short-term or Working Memory (data being used for the task at hand)
Matcher
Conflict Resolver
Long-term Memory (production rules)
Perception
Environment

A production rule architecture

31.2.2 ACT-R

- ACT-R is a cognitive architecture by John Anderson
  - Theory for simulating and understanding human cognition
  - Three main components: modules for action, buffers as interface between modules , and a pattern matcher that matches the current state of buffers


John Anderson

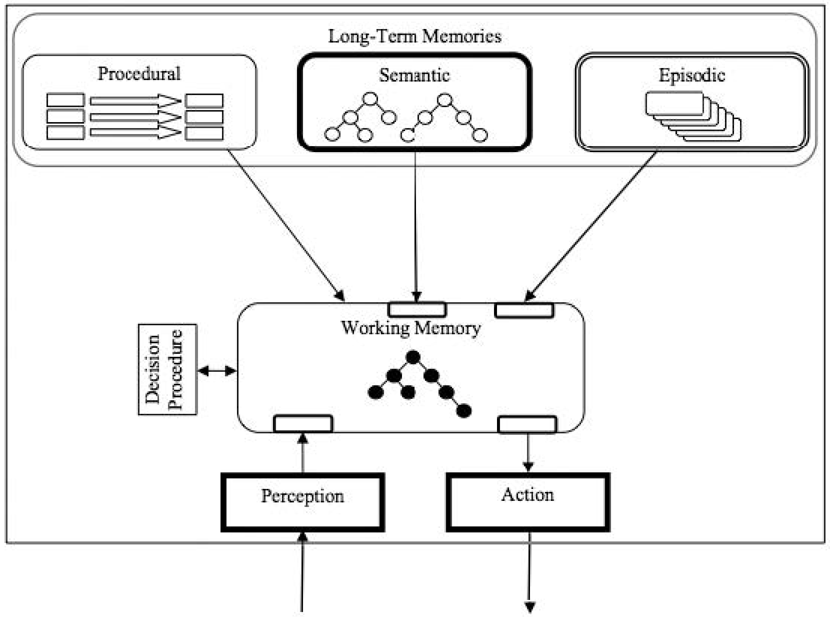


The basic ACT-R architecture

31.2.3 SOAR

- A series of cognitive architecture by John Laird, Allen Newell, and Paul Rosenbloom
  - Acronym for State, Operate And Result (SOAR)
  - The goal for the SOAR is that it serves as a basis for both human and artificial cognition
  - SOAR is something like a programming language having a fixed set of routines.
  - Different kinds of tasks can be programmed in SOAR using these routines

John Laird (left) and Paul Rosenbloom (right).



Memory structures in SOAR