ACT-R 5.0 Architecture

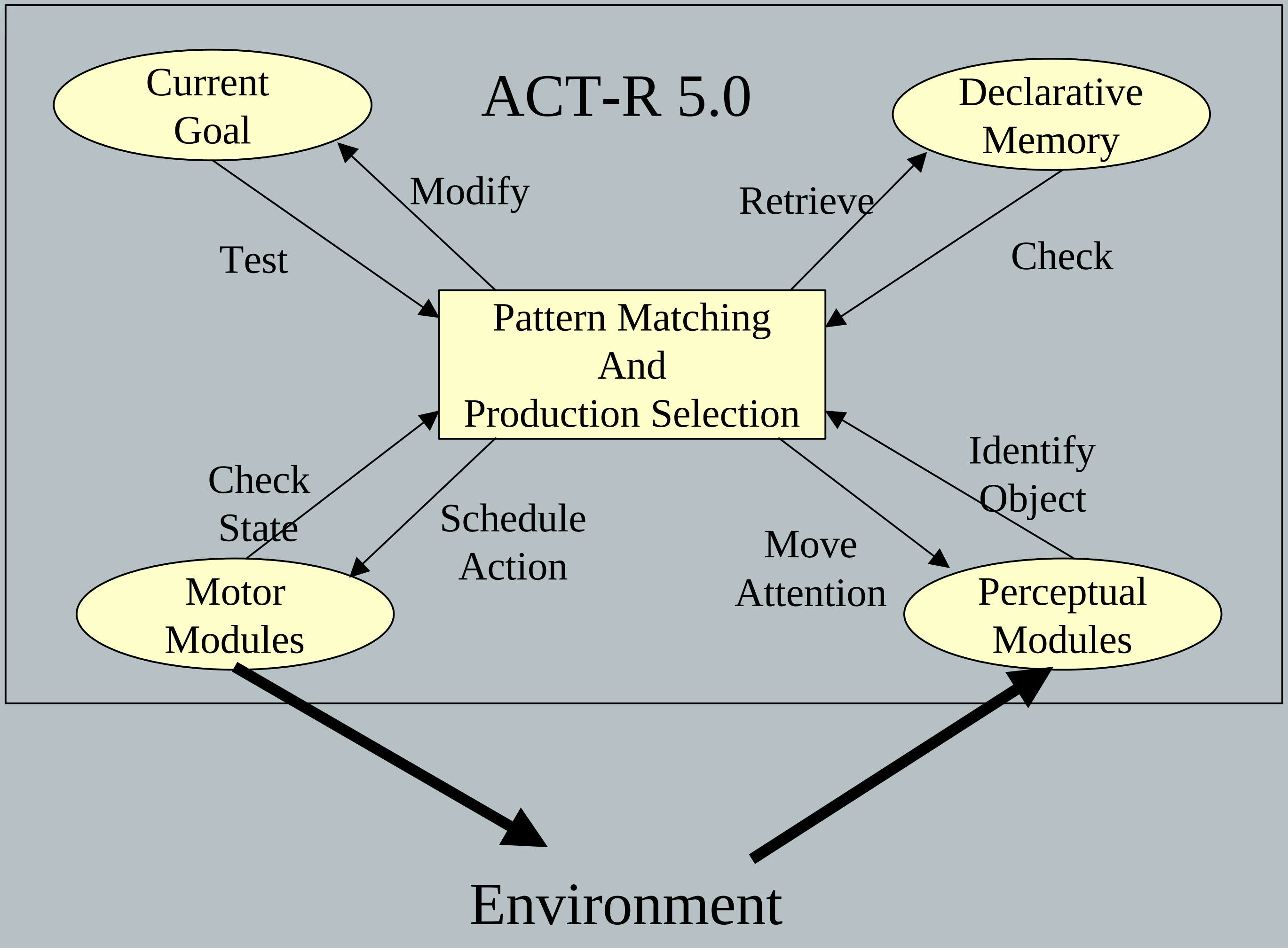
Christian Lebiere

Human-Computer Interaction Institute

Carnegie Mellon University

cl+@cmu.edu

The Architecture

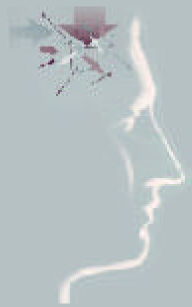


Asynchronous Production Cycle

- Need to be less fixated on the goal and more interruptible
- Asynchronous matching responds to changes and interrupts
- Matches and acts on several sources of information (buffers)
- Any of them can occur in any order or not at all (e.g. no goal)
- What is a buffer? Active part or select location of module?
- Event-centered firing: new cycle as response to buffer change
- Production matching and selection has replaced goal as linchpin
- It also simplifies production cycle since there are no failures
- Need to do something about retrievals that can tie up production matching for several hundred ms and repeatedly during a cycle

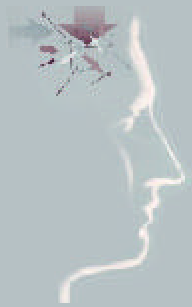
Retrieval as Internal Action

- Separate the request for retrieval from test of results
  - Implausibly long matching cycle and perfect sequential search
  - Unification with treatment of visual perception
  - Better account of neuroscience literature
- Result of retrieval appears in retrieval buffer when complete
- If it fails, a failure (“don’t know”) chunk is put in the buffer
- Retrieval buffer is reset to nil when a new request is made
  - Alternatives: when it is accessed or finite time after available
- Explicit retrieval-state buffer? Busy, time and/or probability?
- Learning (BLL, AL) when and on what basis?
  - Currently: as before at the end of the matching cycle
  - Makes some sense for BLL but how about source changes for AL?



The Buffers

- You can call me -al
- Goal: same but no goal stack.
  - Might be smaller since other buffers might take up some of the load
- Retrieval: holds result of last retrieval.
  - Only 1 chunk, held until next request. Could be more time-limited.
- Visual: holds currently attended visual object.
  - Separate buffers for -location and -state. Could perhaps be integrated.
- Manual: holds last hand command issued.
  - Separate -state buffer to test for jamming.
- Aural: see visual
- Vocal: see manual



The Syntax

- =*buffer*>: Tests against current state of buffer in lhs
  - Modifies chunk in rhs: only kosher for goal, though possible for all
- +*buffer*>: Changes buffer in rhs
  - Goal: new goal equivalent to !focus-on!
  - Retrieval: retrieval request to declarative memory
  - Perceptual/Motor: corresponding request for specific module
- -*buffer*>: Clears buffer in rhs; replaces RPM call to clear
- Summary:
  - Goal: basically unchanged with replacement of stack commands by +
  - Retrieval: separation of retrieval request and retrieval test/binding
  - Perceptual/Motor: replaces ad hoc commands and activation tricks

Example

The Old

```
(p retrieve-sentence-p
   =goal>
      isa recognize-goal
      person =p
      location =l
      pl nil
=prop>
      isa proposition
      relation in*
      arg1 =p
      arg2 =newl
==>
   =goal>
      pp =p
      pl =newl)

(p match
   =goal>
      isa recognize-goal
      location =l
      pl =l
      person =p
      pp =p
==>
   !output! "yes"))
```

The New

```
(p retrieve-sentence-p
   =goal>
      isa recognize-goal
      person =p
==>
   +retrieval>
      isa proposition
      relation in*
      arg1 =p)

(p match
   =goal>
      isa recognize-goal
      person =p
      location =l
   =retrieval>
      isa proposition
       arg1 =p
      arg2  =l
==>
   =goal>
     answer yes
   +goal>
      isa answer-yes)
```

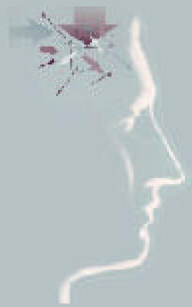
```
Time  0.000: Retrieve-Rule Selected
Time  0.050: Retrieve-Rule Fired
Time  0.354: Rule21 Retrieved
Time  0.354: Instantiate-Rule-0-Args Selected
Time  0.404: Instantiate-Rule-0-Args Fired
Time  0.595: P22 Retrieved
Time  0.595: Ready-To-Read Selected
Time  0.645: Ready-To-Read Fired
Time  0.645: Read-Attend Selected
Time  0.695: Read-Attend Fired
Time  0.695: Module :VISION running command MOVE-ATTENTION
Time  0.745: Module :VISION running command FOCUS-ON
Time  0.745: Read-Bind-Var1 Selected
Time  0.795: Read-Bind-Var1 Fired
Time  1.048: P23 Retrieved
Time  1.048: Retrieve-*Var1-Var2 Selected
Time  1.098: Retrieve-*Var1-Var2 Fired
Time  1.098: Failure Retrieved
Time  1.098: Fail-Harvest-Var Selected
Time  1.148: Fail-Harvest-Var Fired
Time  1.252: Rule25 Retrieved
Time  1.252: Subgoal-Find-Second-Arg Selected
Time  1.302: Subgoal-Find-Second-Arg Fired
Time  1.566: P26 Retrieved
Time  1.566: Ready-To-Read Selected
Time  1.616: Ready-To-Read Fired
Time  5.000: * Running stopped because time limit reached
Time  5.000: Read-Attend Selected
Time  5.050: Read-Attend Fired
Time  5.050: Module :VISION running command MOVE-ATTENTION
Time  5.100: Module :VISION running command FOCUS-ON
Time  5.100: Read-Bind-Var2 Selected
Time  5.150: Read-Bind-Var2 Fired
```

The Semantics

- Buffer tests in lhs take place in parallel
- Buffer actions in rhs take place in parallel
- Any combination of buffers may appear in lhs and rhs
- The whole test-action production cycle takes 50 msec
- Tests take place at the start of the cycle, actions at the end
- Results of actions may not available immediately
- Only 1 retrieval available; a new one clears the previous one
  - Flexible middle ground between full parallelism and blocking action
- No production ever fails: it matches or it does not

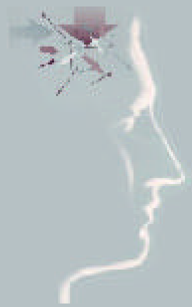
Production Rule Learning

- Need for a more automatic production creation mechanism working directly from execution rather than memory trace
- Works by collapsing consecutive productions together
- Combines buffer tests and actions while avoiding conflicts
- For perceptual/motor buffers, prevent jamming (2 accesses)
- For retrieval buffer, delete request and test when no error
- The resulting production specializes the parent productions
- For goal buffer, merge references unless conflicting changes
- Set parameters to reflect low confidence and cost of entry
- Leave in the more explicit dependency-based mechanism?



Goal Stack

- The goal stack is an idealistic construct
  - Perfect memory
  - Instantaneous retrievals
- Use declarative memory instead
  - Changing to new goal stores previous goal as new memory chunk
  - Old goal can be retrieved from memory (direct link or not?)
  - Recency (and perhaps priming) provides approximation to goal stack
  - Implications of duplicating retrieved goal vs reinstating it directly
- Is goal memory the same as declarative memory or separate?
- No more architectural support for subgoal discounting
- Still a useful abstraction: leave the old commands in place



Miscellaneous

- Outside of the reserved -al keywords, other headers in the lhs are interpreted as retrievals and in rhs as chunk creation (4.0)
- Ultimately: get rid of all !! Commands
- Generalizes negation modifier - to comparison >, <, >=, <=
- New trace format time/event-based instead of cycle-based
  - Cycle trace prints events, latency trace prints durations
- Enable Rational Analysis: Enable Subsymbolic Computations
- Buffers command generalizes goal-focus (and retrieval)
- How central is the goal to cognition given the other sources?
- Issues of compatibility: how to implement the transition?

Open Issues

- What to do if no production matches?
  - Stop: put in a wait production if you want to continue
  - Advance to next cycle and retry: matching and selection can be noisy
  - Advance to next event: need good integrated scheduler
- Find-location (and find-sound): lhs or rhs?
  - Used to be lhs: too powerful because 0-cost matching
  - Currently unified: rhs request and lhs test on visual-location
  - How to handle onset detection: buffer-stuffing of new events
- Questions about direct links, i.e. chunks as slot values
  - Problems with direct retrieval, e.g. competitive latency
  - Problematic direct assignments to buffers: +buffer> =variable
  - Has similar but slightly less power than the goal stack