ACT-R 7 Updates

Dan Bothell

07/26/2017

Last year to now

- 2016 PGSS

  7.0.11-<2054:2016-07-15>

- Now

  7.5-<2244:2017-07-11>

- 5 significant changes
  - possible incompatibilities
- 1 new extra

7.1 & 7.3

- Both similar updates to production compilation
  - Better handling of buffer contexts with respect to strict harvesting
- 7.1
  - Consistent results for original and composed productions as to whether there is or isn’t a chunk in the buffer between and after
- 7.3
  - Pay more attention to the queries

7.1

- If ‘buffer’ is strict harvested and of the goal or imaginal compilation type
- Can’t compose these

```
 (p p1 =buffer> ==>) (p p2 =buffer> ==>)
```

  – The chunk was “stuffed” between them

- Can’t compose these

```
(p p1 =buffer> ==> =buffer> <slot> <value>)
(p p2 =buffer> ==> )
```

  – Empty after p2, but p1&p2 would leave it with a chunk

7.3

- If ‘buffer’ is strict harvested and of the goal or imaginal compilation type
  - Don’t include conditions for queries that are true because of strict harvesting

```
(p p1  =buffer> ==> )
(p p2  ?buffer> buffer empty  ==> )
```

- Result:

```
(p p1&p2 =buffer> ==> )
```

7.2

- Mod-focus schedules it’s change to the chunk instead of directly performing it
  - Consistent with goal-focus

- Allows one to do this:

```
(goal-focus base-goal)
(mod-focus …)
…
```

7.4

- Fixed a bug with RHS !bind! and !mv-bind!
  - Nil return results were being accepted
  - Problem since variables can’t be bound to nil
- Can’t “unselect” the production
  - Print a warning
  - Set the variable’s binding to t

7.5

- Fixed inconsistency with aural-location buffer

- When a sound ends
  - If the corresponding audio-event chunk is in the aural-location buffer
  - Offset and duration slots always updated

7.1.1

- New extra: adaptive-noise
  - Proposed by Christian Lebiere

- Decrease the effect of noise on the activation of chunks as their activation increases with practice

- To use it
  - (require-extra “adaptive-noise”)
  - (sgp :uan t)

Adaptive Noise Mechanism

- Instead of $A = B + S + P + n_{inst} + n_{perm}$

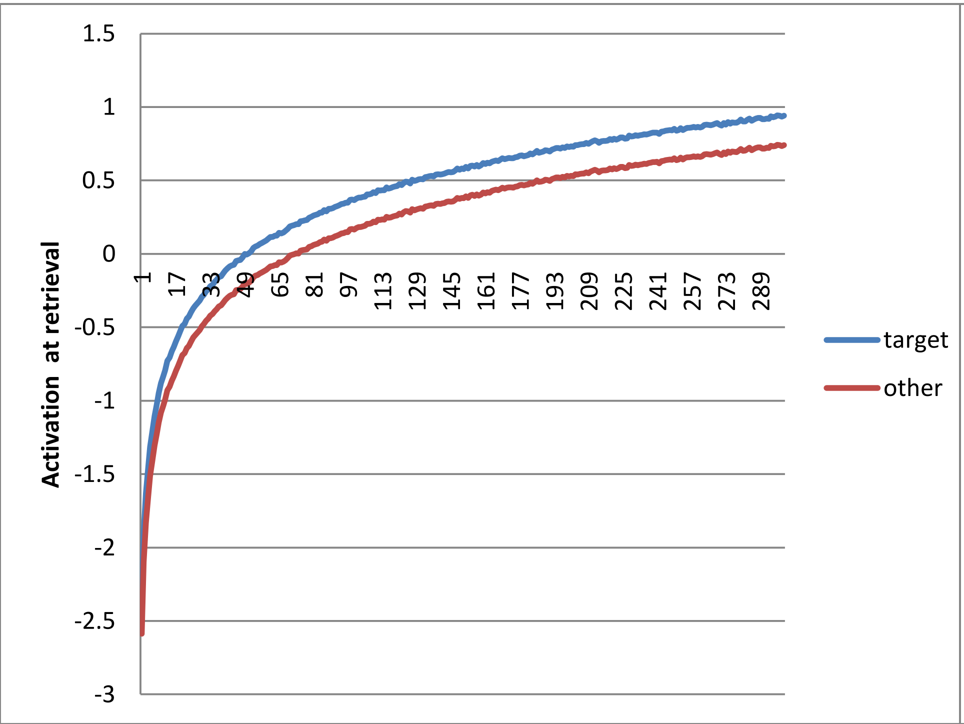
$$A = B' + S + P + n_{perm}$$

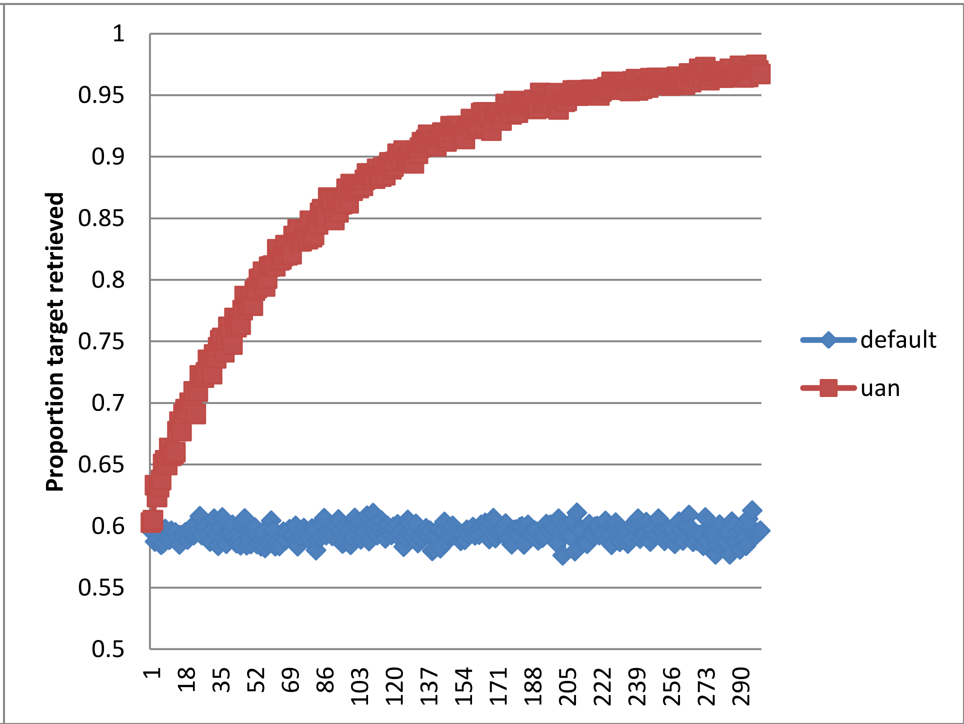
- Where

$$B' = \ln\left(\sum t_i^{-d} + e^{n_{inst}}\right)$$

Effect

- 2 competing chunks e.g. 3+4=7 and 3+5=8
  - – Equal histories
  - – Partial matching in request





ACT-R 7 Status

- Seems to be stabilizing
  - No real mechanism changes
- Maybe it should move to ACT-R 8 soon

New ACT-R Software

Dan Bothell

7/26/2017

Lump of Coal

- ACT-R implemented in Lisp

- Is that really the issue for most?

- Interface for working with ACT-R is Lisp

What is the ACT-R interface

- Commands documented in the manuals

- Lisp REPL and code convenient way to access

- Provide an alternative means of access

Replacement features

- Work with existing ACT-R
- Support all existing documented functionality
- Available via any language
- Allow multiple ‘users’
- Easy to use
- Low overhead cost
- Provide for possible future migration

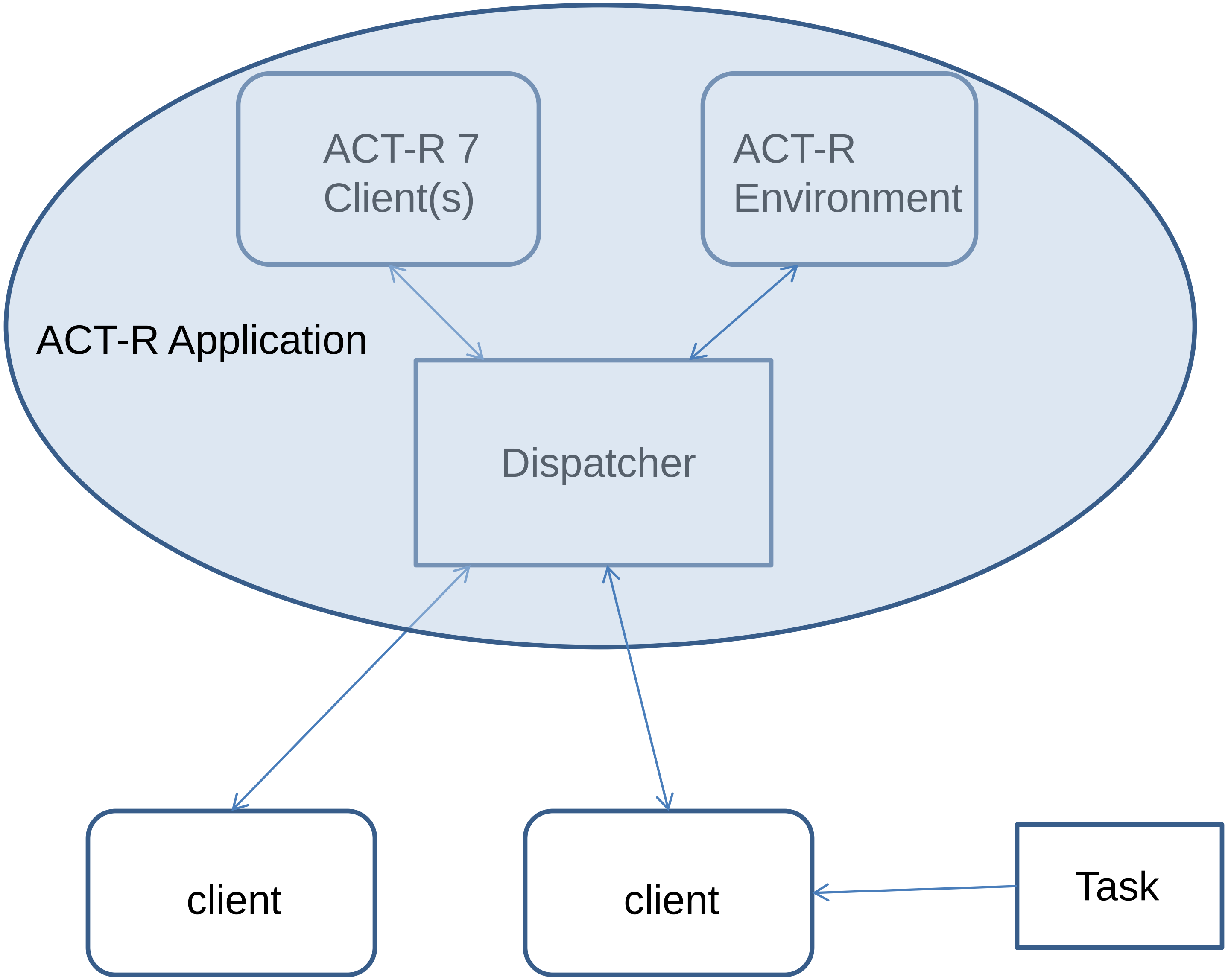
A New Interface

- I’m calling it the dispatcher
- Central server to which clients can connect
- It’s not the user interface
  - System interface
  - Users will interact through something connected to it (clients)

Dispatcher

- A simple RPC server
- Client actions
  - List/check available commands
  - Add/remove a command
  - Evaluate a command
  - Monitor/stop monitoring a command
- Server actions
  - Evaluate a command provided by the client

Overview


ACT-R 7
Client(s)
ACT-R
Environment
ACT-R Application
Dispatcher
client
client
Task

User interaction through a client

- A REPL in a Lisp client
- Python prompt
- MATLAB
- R
- Eclipse
- …

Technical Details

- Connections are TCP/IP socket
  - Dispatcher is listening for connections
- Message protocol is a subset of JSON-RPC 1.0
  - http://json-rpc.org/wiki/specification
  - Not using the class hinting
  - Not using 2.0 since peer<->peer a better fit

Current Status

- Functional
  - Far from complete ACT-R interface
  - significantly underdocumented
  - Environment still connected directly to ACT-R
- Only source code right now
  - Requires Quicklisp
- Available from the ACT-R repository svn://act-r.psy.cmu.edu/actrDES

- Also includes a Python client and interface library
  - All tutorial *tasks* have been reimplemented completely in Python

Tasks

- Models are separate from tasks
  - One model file (.lisp)
  - Separate .lisp and .py task files
- Uses the AGI not native GUI

- Unit 5 1-hit-blackjack
  - Similarity hook in the task code
- Unit 6 BST
  - Math still in the model's Lisp

    !eval! (< =under (- =over 25)))
- Unit 5 grouped example

  !eval! ("grouped-response" =name)

Relative to the desired features

- Good
  - Allow multiple ‘users’
  - Available via any language
  - Provide for possible future migration
  - Easy to use

- Mixed
  - Work with existing ACT-R
  - Low overhead cost
  - Support all existing documented functionality

Work with existing ACT-R

- Tutorial models almost completely unchanged

- Multiple clients requires better concurrency safety than existing ACT-R has (i.e. none)
- More safety checks on parameters
- Lots more error protection

Low overhead cost

- Not horrible for tested tutorial tasks
  - Compared ACT-R 7 to using Python client
  - Range from ~1% faster to ~600% slower
  - Amount of back and forth a factor
  - Safety code has a noticeable cost
- Mostly focused on functionality at this point
  - Added a switch for the similarity hook to cache values instead of calling out each time
  - Hope to get some reasonable gains by running modules in parallel

Support all existing documented functionality

- Keyword parameters and chosen RPC protocol not a good fit
  - Use optional params and/or simplified alternate functions instead
- “Wrapper” macros like no-output not feasible
  - Replace with separate off/on calls
- RPM device mechanism doesn’t really work
  - Replaced it with individual module specific interfaces
  - Use the dispatcher monitoring to respond to actions
  - Embrace that chunk-types are unnecessary for visual features

Next Steps

- Additional safety code around currently provided functionality
- Update tutorial documentation to include information on both the Lisp and Python task implementations
- Have the Environment work through the dispatcher