Computing Dynamic Meanings: Building Integrated Competence-Performance Theories for Semantics

Day 1, part 2: pyactr tutorial

Jakub Dotlačil & Adrian Brasoveanu

ESSLLI 2018, August 6 2018
Plan

- symbolic system
- simple agreement model
- environment module
- counting model
Install pyactr:

- use pip, or
- go to github:
  https://github.com/jakdot/pyactr
Basic workflow in (py)actr

1. create chunks
2. create model
   ▶ store chunks in declarative memory
   ▶ (create extra modules/buffers)
   ▶ create rules
   ▶ (create environment process)
3. run the model with parameters of interest
4. read off behavioral data from the simulation
1. John definitely [CONCEPT SLEEP].

- The code for this model is available at:
  https://people.ucsc.edu/~abrsvn/esslli-2018-day1.html
- example1.py
Agreement model I; creating chunks

```python
import pyactr as actr

#chunktypes
actr.chunktype("word", "form category meaning
    number person function")

actr.chunktype("concept", "meaning")
```
Agreement model I; creating chunks

chunk_john = actr.chunkstring(string=""
  isa          word
  form         john
  category     noun
  meaning      john
  number       sg
  person       3
  function     subject"")

chunk_definitely = actr.chunkstring(string=""
  isa          word
  form         definitely
  category     adverb
  meaning      definitely
  function     speaker_adverb"")
Agreement model I; creating chunks

chunk.sleeps = actr.chunkstring(string=""
    isa       word
    form      sleeps
    category  verb
    meaning   sleep
    number    sg
    person    3
    function  predicate"")

chunk_concept.sleep = actr.chunkstring(string=""
    isa       concept
    meaning   sleep"")
Agreement model I; creating chunks

\texttt{print}(chunk\_definitely < chunk\_john)
False

\texttt{print}(chunk\_john < chunk\_john)
False

\texttt{print}(chunk\_john \leq \text{ chunk\_john})
True

\texttt{print}(chunk\_sleeps < chunk\_concept\_sleep)
False

\texttt{print}(chunk\_concept\_sleep < chunk\_sleeps)
True

\texttt{print}(chunk\_concept\_sleep \geq \text{ chunk\_sleeps})
False
Agreement model I; creating chunks

```python
actr.chunktype("read", "task current_word")

starting_goal = actr.chunkstring(string=""
    isa read
    task speaking
    current_word None"")
```
Agreement model II; creating model

#II: create model

agreement = actr.ACTRModel()
Agreement model III; store chunks in the declarative memory

#III: store chunks in the decl. memory and buffers

agreement.decmem.add(chunk_john)
agreement.decmem.add(chunk_definitely)
agreement.decmem.add(chunk_sleeps)

agreement.goal.add(starting_goal)
Agreement model IV; create extra modules

agreement.set_goal(name="imaginal", delay=0.05)

agreement.goals["imaginal"].add(chunk_concept_sleep)
Agreement model V; create productions

agreement.productionstring(name="match current word", string=""
  =g>
  isa read
  task speaking
  =imaginal>
  isa concept
  meaning sleep
  ==> 
  =g>
  isa read
  task recalling_subject +retrieval>
  isa word
  category noun
  function subject
  """)
Agreement model V; create productions

```python
agreement.productionstring(name="agree", string=""
  =g>
  isa    read
  task   recalling_subject
  =imaginal>
  isa    concept
  meaning=x
  =retrieval>
  isa    word
  category=noun
  function=subject
  number=n
  ==>
  =g>
  isa    read
  task   recalling_verb
  +retrieval>
  isa    word
  category=verb
  meaning=x
  number=n
  ""
)```
Agreement model V; create productions

```python
agreement.productionstring(name="done", string=""
    =g>
    isa          read
    task         recalling_verb
?retrieval>
    state        free
    =retrieval>
    isa          word
    =>
~imaginal>
    =g>
    isa          read
    task         done
    current_word =retrieval ""
)
Agreement model VI; run the model

agreement_sim = agreement.simulation()
agreement_sim.run()
Read off behavioral data from the simulation

- example2.py
Create environment process

- example3.py

Counting
Counting + environment

```python
import pyactr as actr
actr.chunktype("counting", "state counted end")
environment = actr.Environment(focus_position=(20, 20))
counter = actr.ACTRModel(environment)
counter.goal.add(actr.chunkstring(name="reading", string=""
    isa     counting
    state   start
    counted 0
    end     3""""))
```
Counting + environment, rules

counter.productionstring(name="move attention", string=""

=g>
isa counting
state start
?visual_location>
buffer full
=visual_location>
isa _visuallocation
?visual>
buffer empty
state free
==>
=g>
isa counting
state encode
~visual_location>
+visual>
isa _visual
cmd move_attention
screen_pos =visual_location"")
Counting + environment, rules

counter.productionstring(name="encode first letter", string="""
    =g>
    isa counting
    state encode
    counted 0
    =visual>
    isa _visual
    value ~None
    ==>
    ~visual>
    =g>
    isa counting
    state search
    counted 1
    """
)
Counting + environment, rules

counter.productionstring(name="find_probe", string=""
  =g>
  isa counting
state search
?visual_location>
buffer empty
==> 
=g>
=g>
isa counting
state start
?visual_location>
attended False
+visual_location>
isa _visuallocation
screen_x lowest
screen_y closest""")
counter.productionstring(name="stop", string="""
  =g>
  isa    counting
  state  search
  counted =x
  end    =x
  ==>    
  ~g>    
  """
)
Exercises

1. Counting up to 5
2. ‘most’ (e.g., ‘most letters are As’)