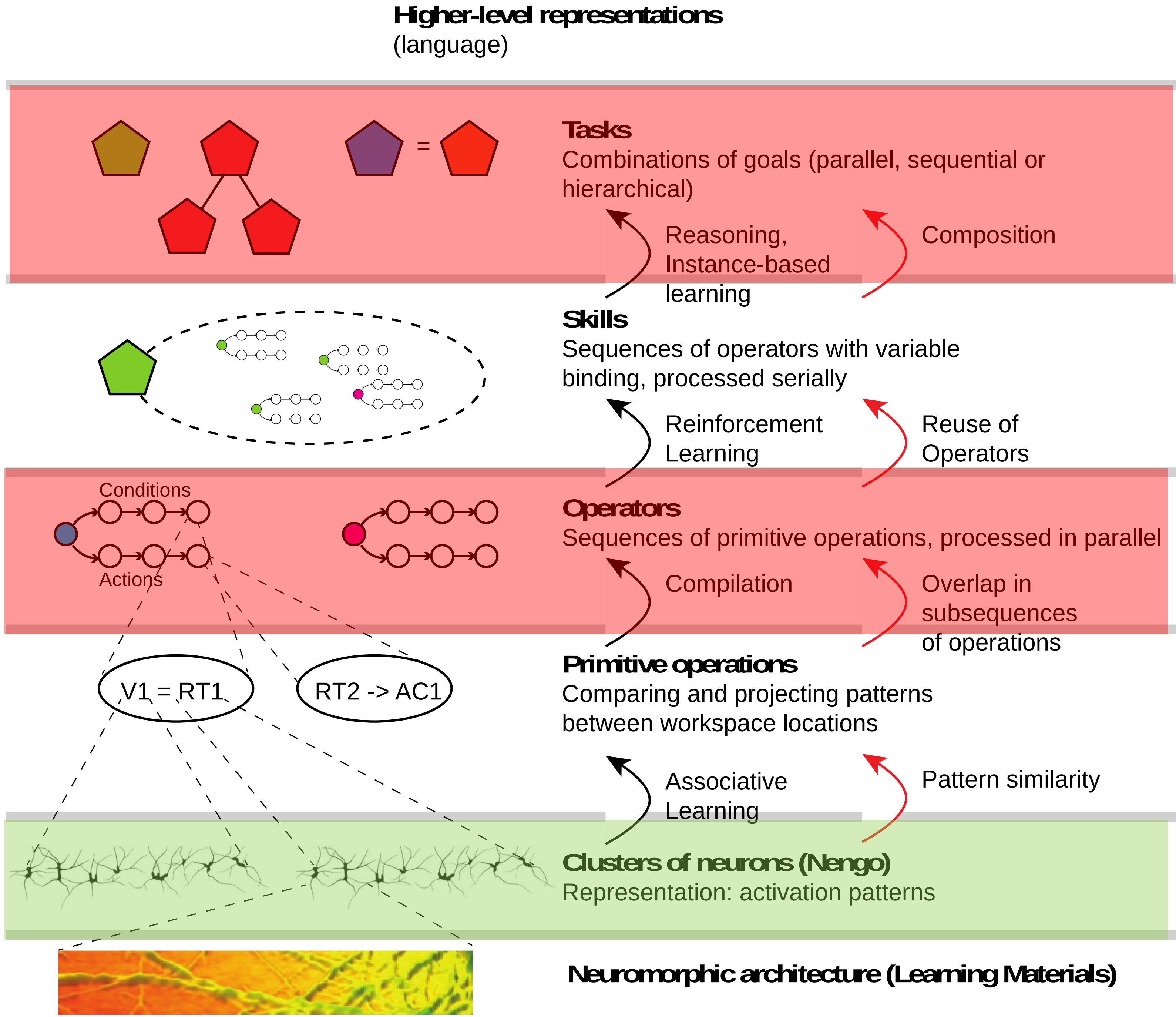
Extending ACT-R’s Modeling Capabilities: One Level Below, and One Level Above

Niels Taatgen

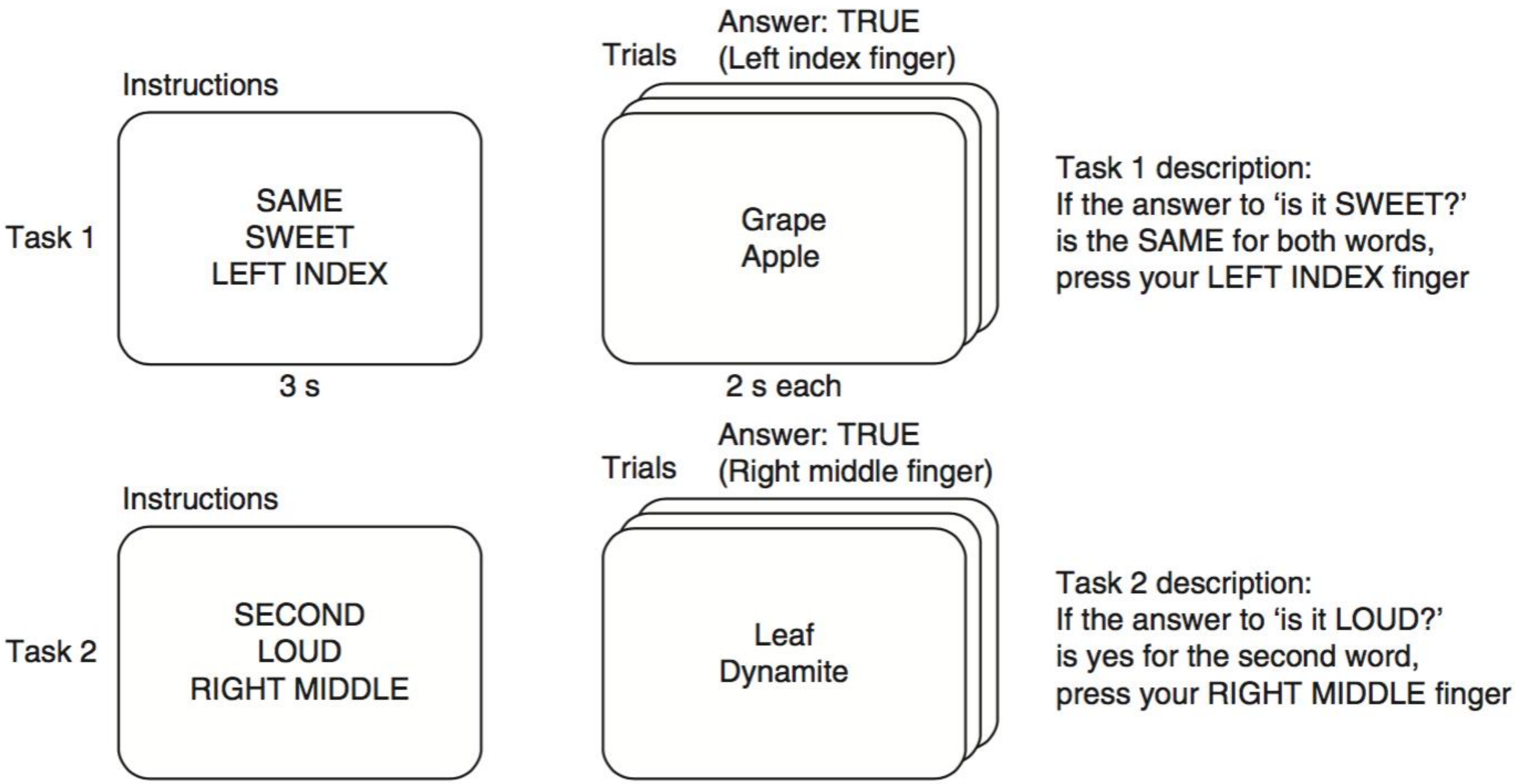
Bernoulli Institute, Artificial Intelligence

University of Groningen



Higher-level representations
(language)
Tasks
Combinations of goals (parallel, sequential or hierarchical)
Reasoning, Instance-based learning
Composition
Skills
Sequences of operators with variable binding, processed serially
Reinforcement Learning
Reuse of Operators
Conditions
Actions
Operators
Sequences of primitive operations, processed in parallel
Compilation
Overlap in subsequences of operations
V1 = RT1
RT2 -> AC1
Primitive operations
Comparing and projecting patterns between workspace locations
Associative Learning
Pattern similarity
Clusters of neurons (Nengo)
Representation: activation patterns
Neuromorphic architecture (Learning Materials)

Example: RITL experiment



Just One

Loud

Left Middle

+

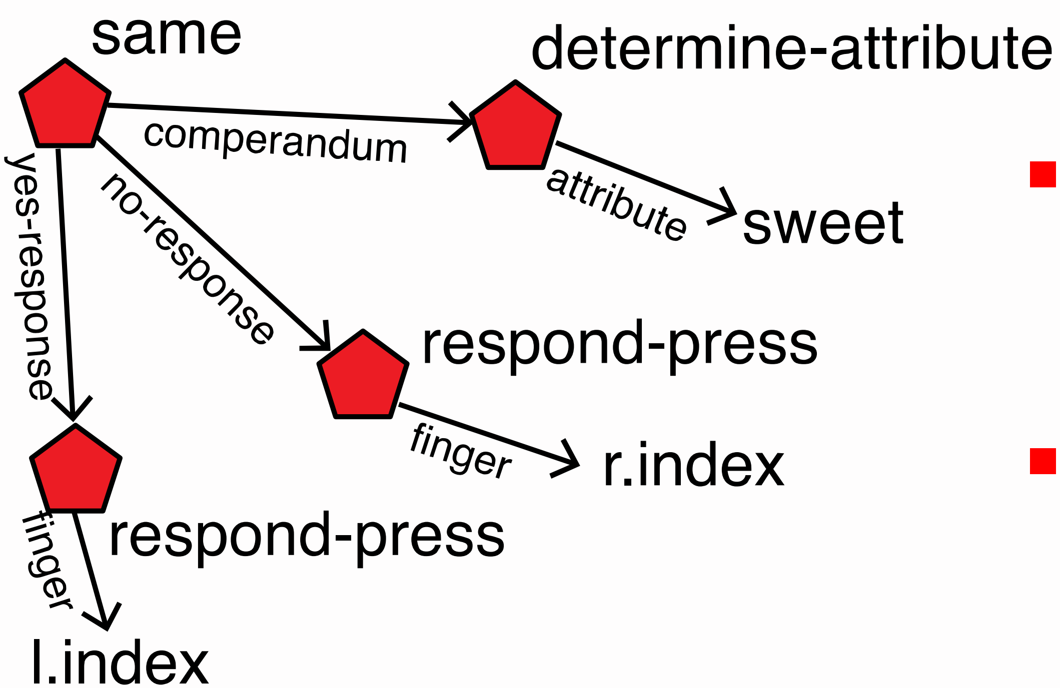
Drums

Airplanes

What skills do we need?

- Skills for same/justone/second/not second
- A skill that tests an attribute that we can instantiate
- A skill that pushes buttons that we can instantiate

Skill level model



- Primitives at the task level are skills
- Learning by a (linguistic) composition
- One-shot-learning!

We can compose these using chunks

just-one-1
- slot1 just-one
- arg-subskill-1 determine-attribute-1
- success-skill press-finger-yes
- fail-skill press-finger-no

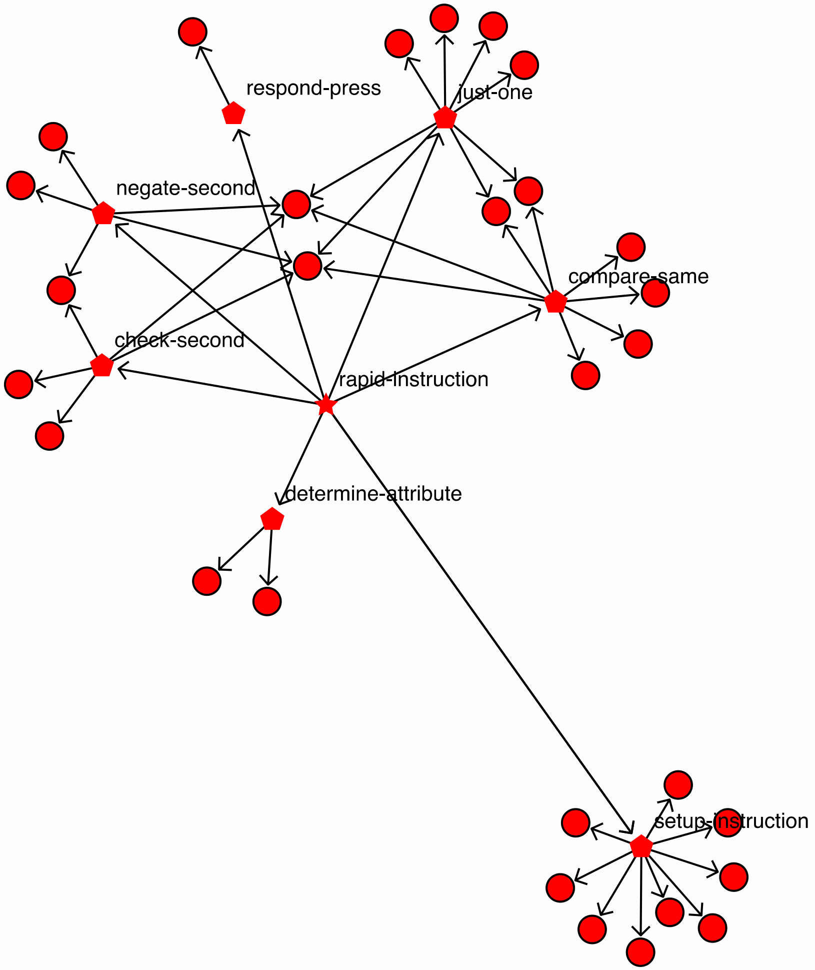
determine-attribute-1
- slot1 determine-attribute
- fact-type loud

press-finger-yes
- slot1 press-finger
- finger left-index

press-finger-no
- slot1 press-finger
- finger right-index

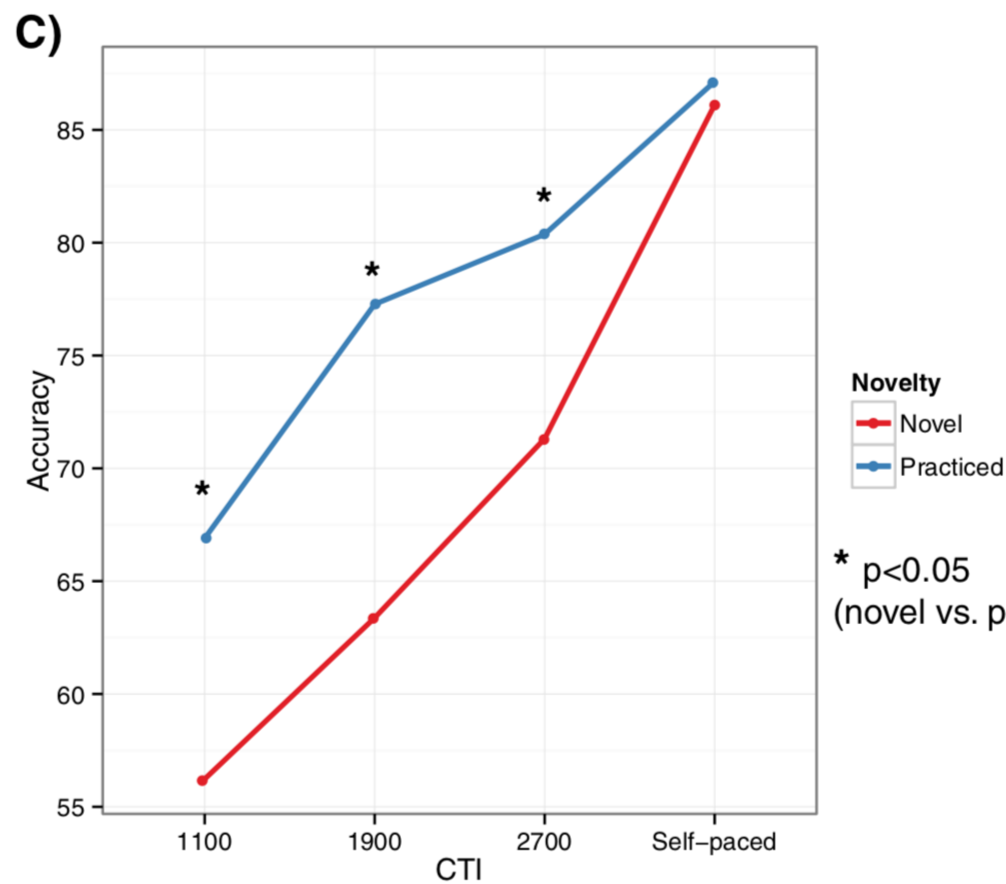
The complete model

- Implementation of component skills: same, just-one, etc.
- A skill that builds the declarative structure that serves as a goal representation



Cole et al. 2017 experiment

- Limited time to study instruction (1100, 1900, 2700ms)
- Response within 1500ms
- Practiced vs. novel instructions



Model challenges

- How to get the model fast enough to be able to do this at all within the limited time available?
- How to model the difference between trained and novel?

Instruction parsing needs to generate four chunks

just-one-1
- slot1 just-one
- arg-subskill-1 determine-attribute-1
- success-skill press-finger-yes
- fail-skill press-finger-no

determine-attribute-1
- slot1 determine-attribute
- fact-type loud

press-finger-yes
- slot1 press-finger
- finger left-index

press-finger-no
- slot1 press-finger
- finger right-index

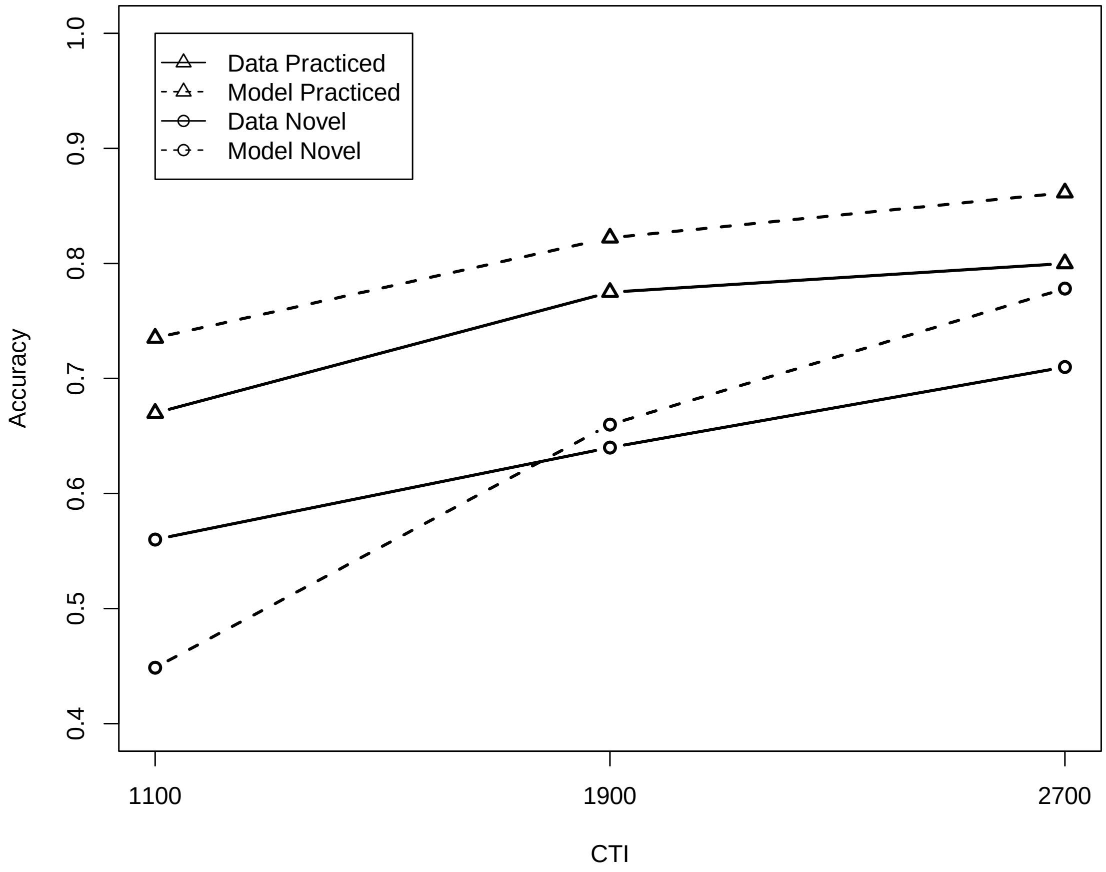
Retrieval?

- If the instruction is “just-one loud left-index” again, you cannot just retrieve that
- But we can retrieve based on one (just-one), and hope for the best
- If we retrieve something else, we will modify it to fix it

Procedure

- Try to retrieve based on the logical cue
  - If this fails, build a new chunk
- For the subskill, success-skill and fail-skill
  - First check whether the skill that is already there is correct
  - If not, try to retrieve the correct skill to replace it
  - If that fails, build it from scratch

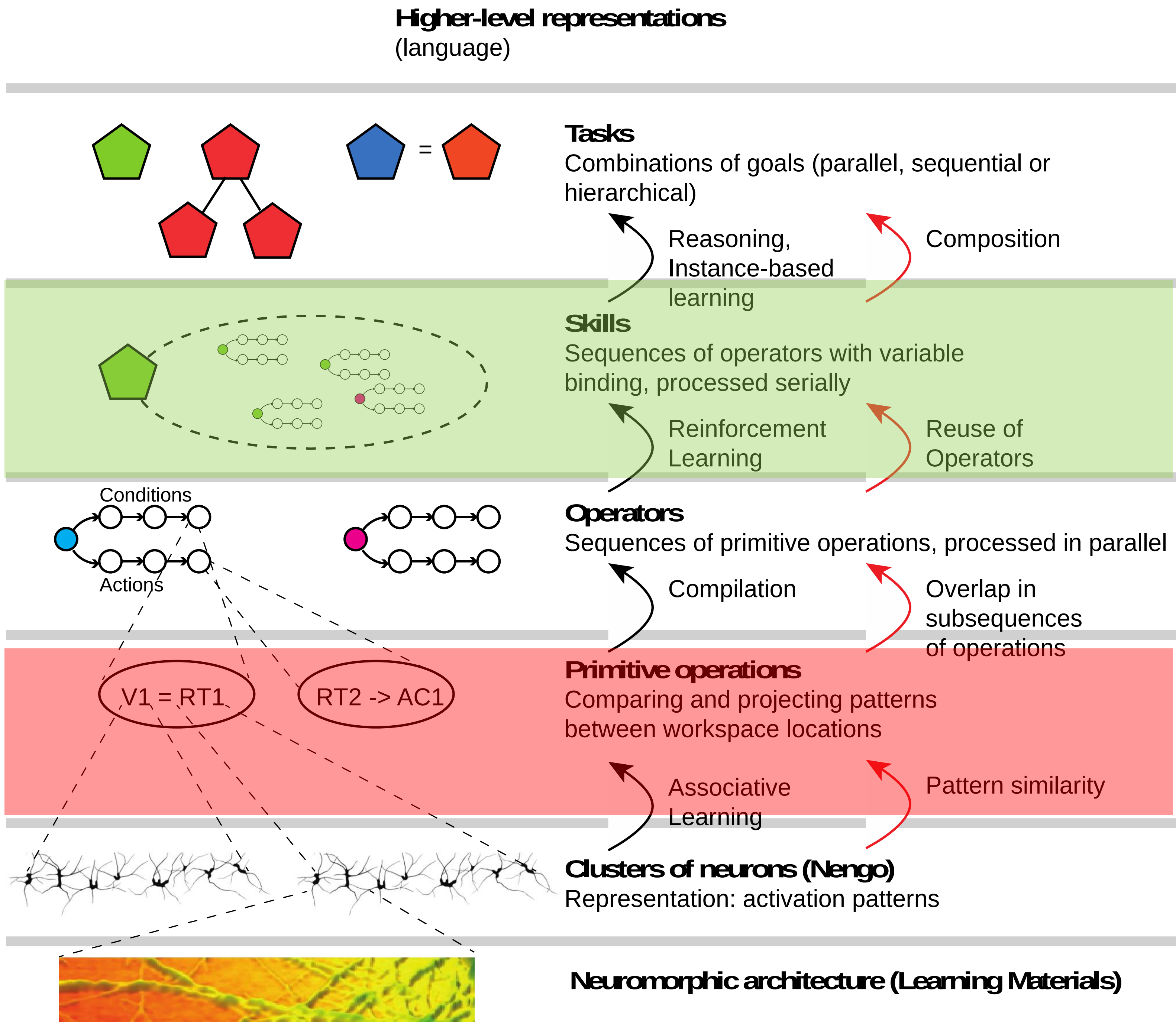
Results



Data Practiced
Model Practiced
Data Novel
Model Novel
Accuracy
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1100
1900
2700
CTI

More general agenda

- Can we start building a general purpose skill library that can serve as a basis for more constrained modeling?
- If we have that, we can build instantiations of the cognitive architecture that already have a rich skill set



Higher-level representations
(language)
Tasks
Combinations of goals (parallel, sequential or hierarchical)
Reasoning,
Instance-based
learning
Composition
Skills
Sequences of operators with variable binding, processed serially
Reinforcement
Learning
Reuse of
Operators
Conditions
Actions
Operators
Sequences of primitive operations, processed in parallel
Compilation
Overlap in
subsequences
of operations
V1 = RT1
RT2 -> AC1
Primitive operations
Comparing and projecting patterns between workspace locations
Associative
Learning
Pattern similarity
Clusters of neurons (Nengo)
Representation: activation patterns
Neuromorphic architecture (Learning Materials)

Lower level of abstraction

- How can we learn operators from primitive operations?
  - Construct basic operators with a single primitive operation
  - Use “operator compilation” and reinforcement learning to discover the knowledge needed for a task

Example: Choice-reaction task

```
define facts {
        (f1 associate vanilla thumb)
        (f2 associate ice ring)
        (f3 associate paper pinkie)
        (fa1 taste vanilla sweet)
        (fa2 temperature ice cold)
        (fa3 weight paper small)
        (fb1 color vanilla yellow)
        (fb2 texture ice slippery)
        (fb3 color paper white)
}
```

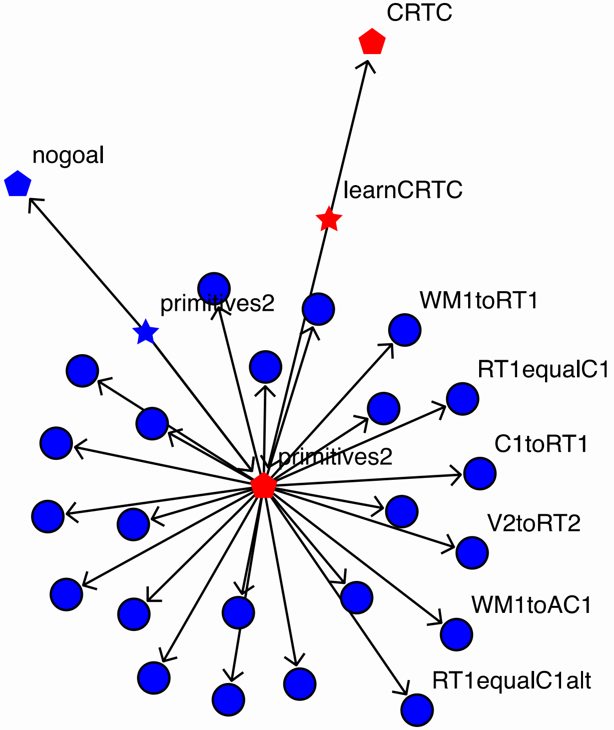
Example basic operators

```
operator V1toRT2 {
    V1 <> nil
==>
    V1 -> RT2
    nil -> V1
}
```

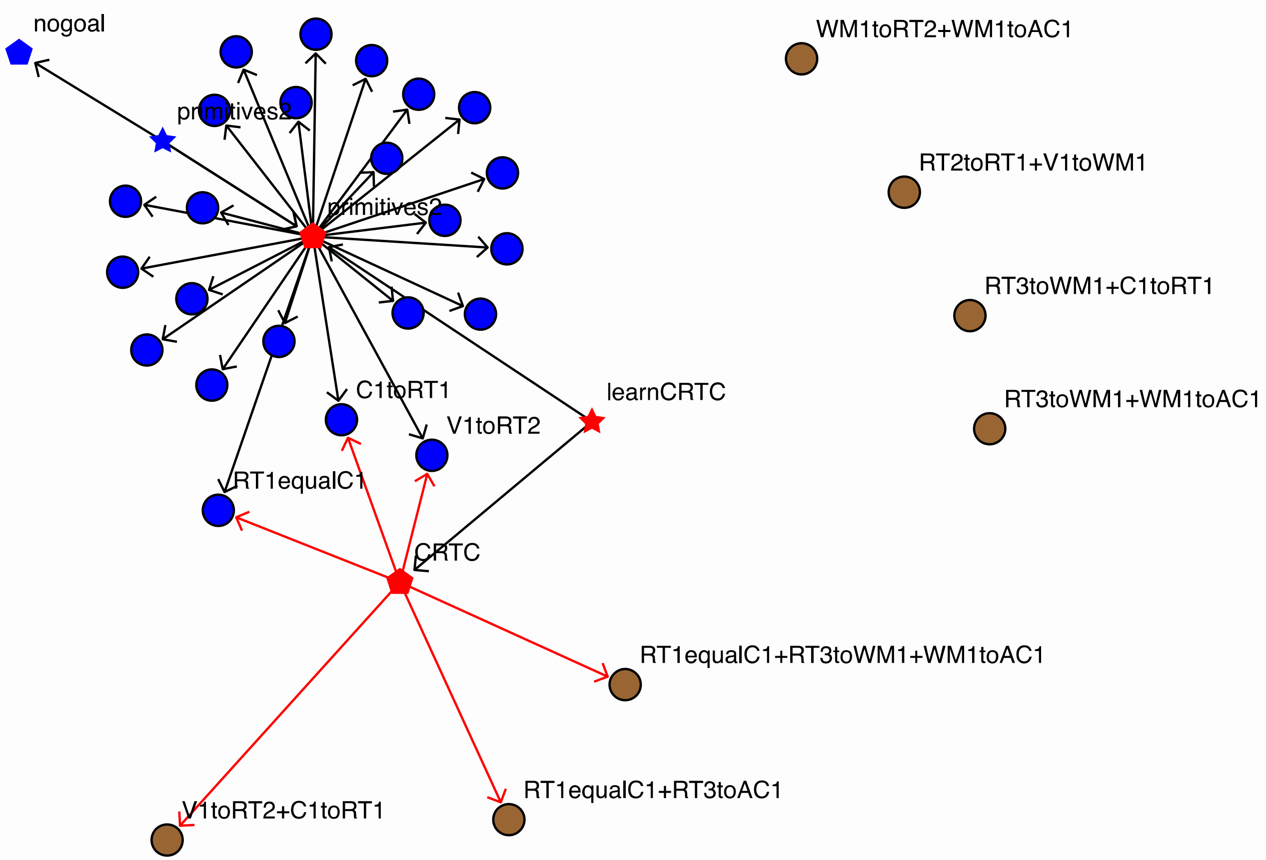
```
operator RT2toRT1 {
        RT2 <> nil
==>
        RT2 -> RT1
}

operator RT1equalC1 {
        RT1 = *fact-type
==>
}
```

Initial model



After learning



One possible solution

Operator 1

V1 <> nil

==>

V1 -> RT2

associate -> RT1

Operator 2

RT1 = associate

RT3 <> nil

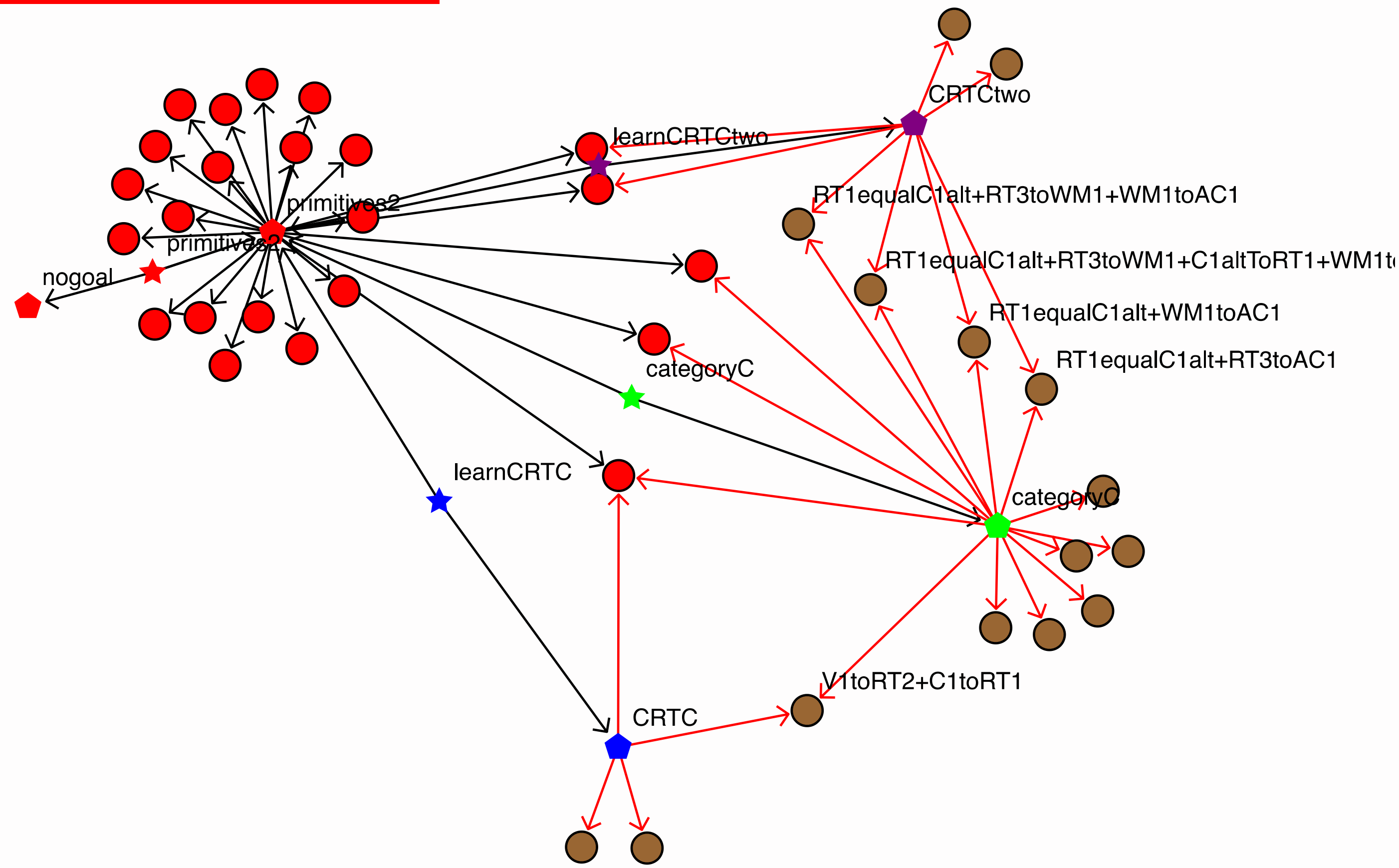
==>

action -> AC1

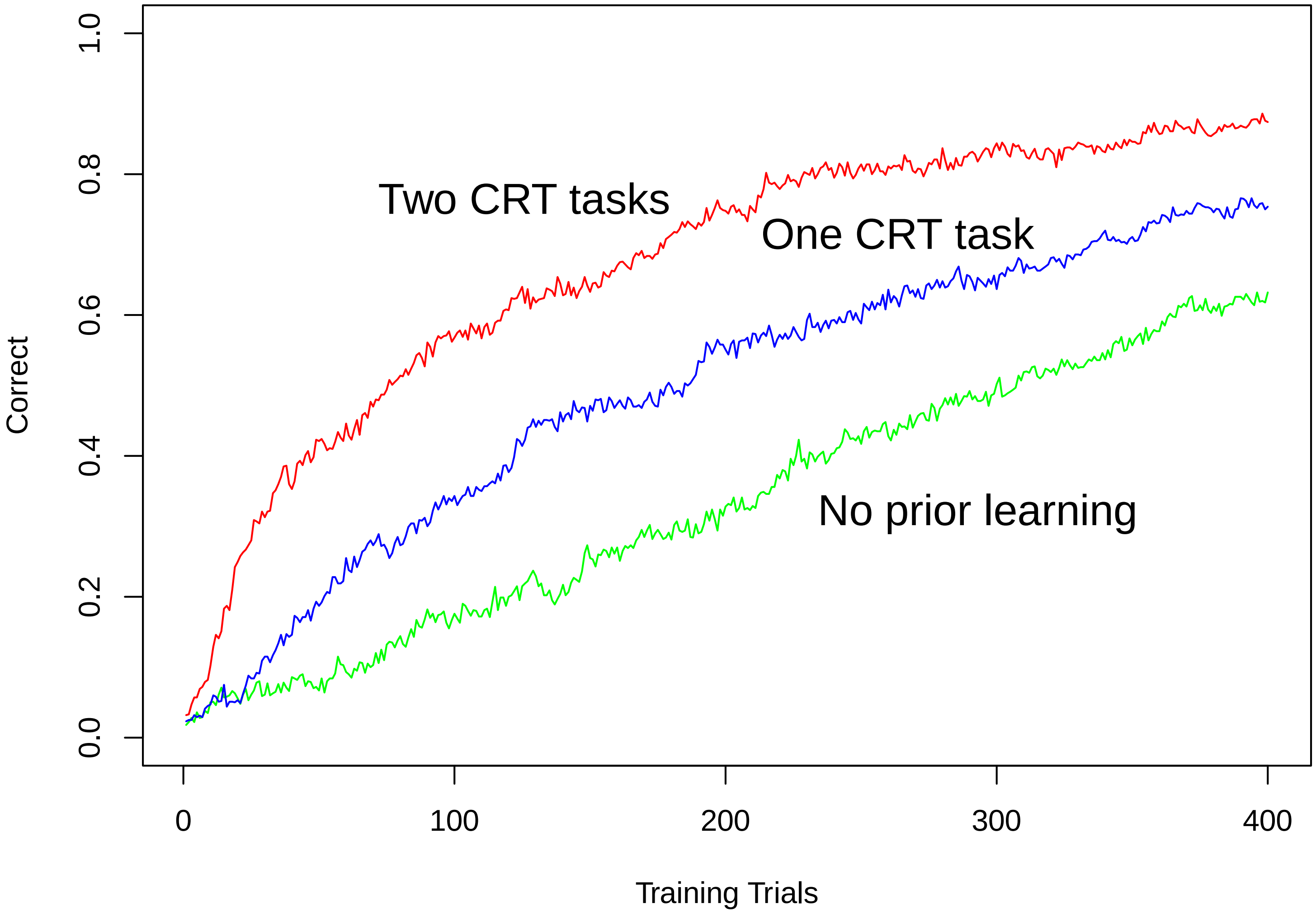
RT3 -> AC2

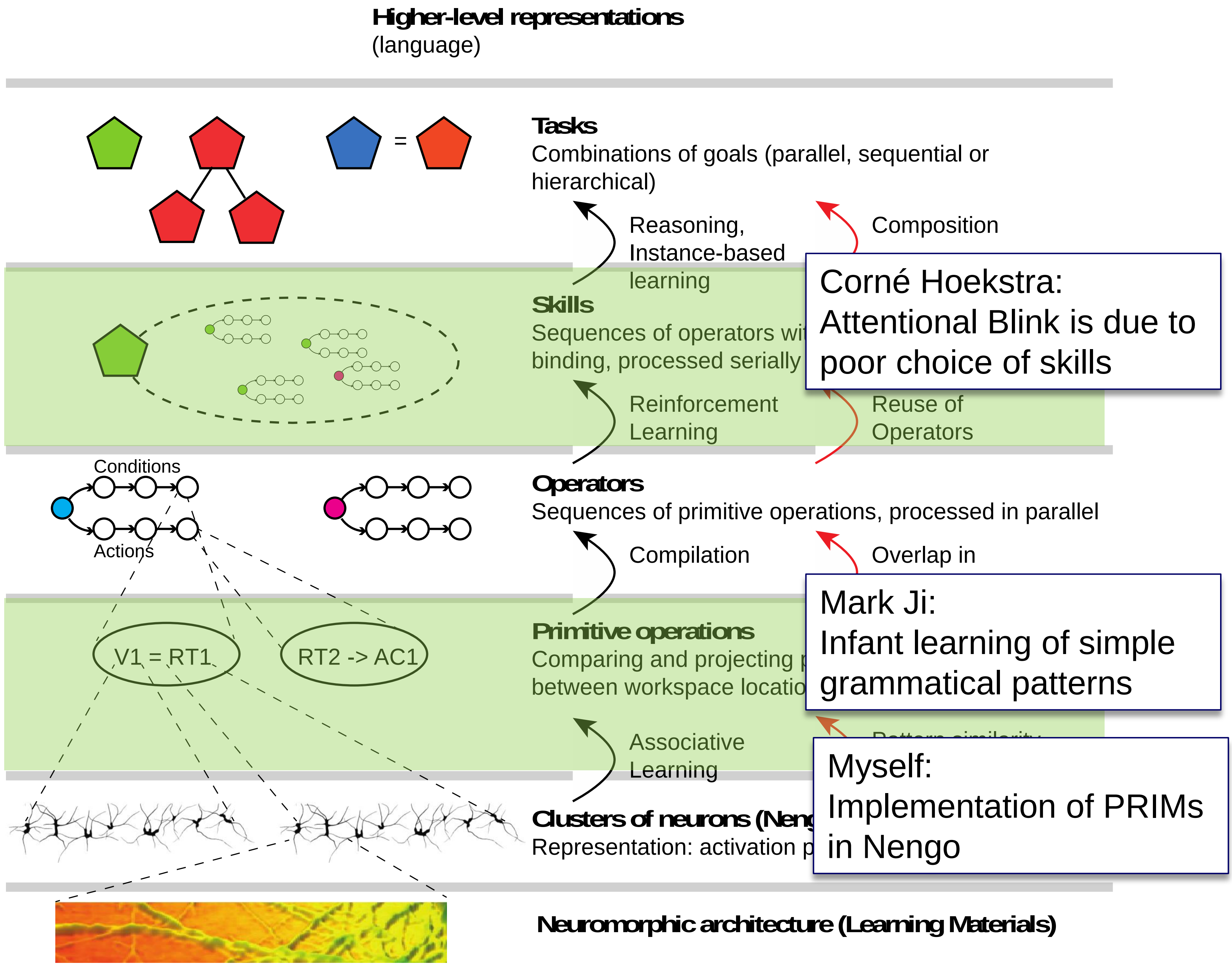
More complex task

```
define facts {
        (fac1 category goat animal)
        (fac2 category pinguin animal)
        (fac3 category cabbage plant)
        (fac4 category tulip plant)
        (altfac1 property goat hair)
        (altfac2 property pinguin notfly)
        (altfac3 property cabbage food)
        (altfac4 property tulip mania)
        (fca1 response animal left)
        (fca2 response plant right)
        (fca3 response somethingelse middle)
        (fca4 response differentyet upper)
        (fca5 response noguessing lower)
        (altfca1 property animal livingthing)
        (altfca2 property plant livingthing)
}
```


primitives2
primitives2
nogoal
learnCRTCtwo
CRTCtwo
RT1equalC1alt+RT3toWM1+WM1toAC1
RT1equalC1alt+RT3toWM1+C1altToRT1+WM1t
RT1equalC1alt+WM1toAC1
RT1equalC1alt+RT3toAC1
categoryC
categoryC
learnCRTC
V1toRT2+C1toRT1
CRTC

Learning is more successful with prior learning





Higher-level representations
(language)
Tasks
Combinations of goals (parallel, sequential or hierarchical)
Reasoning, Instance-based learning
Composition
Corné Hoekstra:
Attentional Blink is due to poor choice of skills
Skills
Sequences of operators wi
binding, processed serially
Reinforcement Learning
Reuse of Operators
Conditions
Actions
Operators
Sequences of primitive operations, processed in parallel
Compilation
Overlap in
Mark Ji:
Infant learning of simple grammatical patterns
V1 = RT1
RT2 -> AC1
Primitive operations
Comparing and projecting
between workspace locatio
Associative Learning
Myself:
Implementation of PRIMs in Nengo
Clusters of neurons (Neng
Representation: activation p
Neuromorphic architecture (Learning Materials)