Learning a Complex Dynamic Skill

John R Anderson
Dan J. Bothell
Scott A. Douglass
Craig Haimson

Bragging Points:
1. Relatively complex, realistic, dynamic task.
2. All the parameters turned on.
3. Learning from actual instruction and production compilation really working.
4. Fitting data in detail.
5. On the way to a synthetic student.
Instructions for CMU-ASP

1. The task is to identify unidentified tracks. Unidentified tracks are half squares with vectors emanating from them. One should hook (click on) such tracks and then go through the sequence of identifying them. (To identify-tracks first look-for a track that is "half-square" then hook the track then idsequence the track and then repeat)

2. One way to identify a track is to confirm that it is flying at a commercial altitude and speed and then record it as friendly primary id and non-military air id. (To idsequence a track first altitude-test then speed-test and then record it as "friend" "non-military")

3. The other way to identify a track is to request its EWS identity, and then classify the track according to that identity. (To idsequence a track first ews the track for a ews-signal and then classify the track according to the ews-signal)

4. To confirm that a plane is flying at the commercial altitude, look in the upper left, search down for “alt”, read the value to the right, and confirm that it is more than 25,000 and less than 35,000. (To altitude-test first seek "upper-left" and then search-down for "alt" at a location then read-next from the location a value then check-less 25000 than the value and then check-less the value than 35000)
5. To confirm that a plane is flying at the commercial speed, look in the upper left, search down for “speed”, read the value to the right, and confirm that it is more than 350 and less than 550. (To speed-test first seek "upper-left" and then search-down for "speed" at a location then read-next from the location a value then check-less 350 than the value and then check-less the value than 550)

6. To request the EWS identity of a track, select the “ews” key, then select “query sensor status” key, and encode the value that you are told. (To ews a track for a ews-signal first select "ews" then select "query sensor status" and then encode-ews the ews-signal)

7. To classify a track whose EWS identity is ARINC record it as “friendly” primary id and “non-military” air id. (To classify a track according to a ews-signal first match the ews-signal to "arinc564" and then record it as "friend" "non-military")

8. To classify a track whose EWS identity is APQ record it as hostile primary id and strike air id. (To classify a track according to a ews-signal first match the ews-signal to "apq" and then record it as "hostile" "strike")

9. To classify a track whose EWS identity is APG record it as friendly primary id and strike air id. (To classify a track according to a ews-signal first match the ews-signal to "apg" and then record it as "friend" "strike"
Instructions for CMU-ASP

10. To classify a track whose EWS identity is negative treat it as unclassifiable. (To classify a track according to a ews-signal first match the ews-signal to "negative" and then mark-node the track)

11. To record a primary id and a secondary id select the following sequence of keys: “track manager”, “update hooked track”, “class/amp”, “primary-id”, the primary id, “air-id”, the air-id, “save” and then you have succeeded. (To record a primary-id and a air-id first select "track manager" then select "update hooked track" then select "class/amp" then select "primary id" then select the primary-id then select "air id amp" then select the air-id then select "save changes" and then success)

12. To select a key, find where it is in the menu and hit the corresponding F-key. (To select a option first find-menu the option at a location and then hit-key corresponding to the location.

13. To find where an item is in the menu, look to the lower left and search to the right for the term. (To find-menu a option at a location first seek "lower-left" and then search-right for the option at a location)
The Basic Plan for Learning from Instruction

- Instructions are encoded as declarative structures characterizing the sequence of goals that must be achieved.
- There are a set of production rules that will interpret any such sequence of instructions.
- Production compilation will convert this into task specific procedures.
- As an aside we solve the mystery of task instructions that has haunted Experimental Psychology.
**Production Compilation: Applied to CMU-ASP**

IF trying to retrieve a rule to achieve a goal and rule for achieving that goal has been retrieved THEN retrieve the first step of that rule and note trying to recall the first step.

The first step in recording an id is to select “track”.  

IF trying to retrieve the first step of a goal and a step has been retrieved involving a subgoal THEN change goal to trying to achieve that subgoal and try to retrieve a rule to achieve that subgoal.

Results in:

IF trying to retrieve a rule for recording an id THEN set a subgoal to select “track” and try to retrieve rule for selecting “track”.
Eventually production rules are learned like:

IF trying to retrieve a rule for recording an id
THEN set a hit F1
    and set a subgoal to select “update”

Goes no further because of motor jamming

The model moves from taking almost 100 seconds to classify a plane to less than 10 seconds. Part of the learning depends on production compilation and part of it depends on location learning.
Niels Taatgen’s Subsymbolic Model

1. Allows for more gradual introduction of rules
2. Based on Rescorla-Wagner Rule
3. More robust across a range of applications

\[
EG = \frac{n \cdot priorEG + m \cdot ExperiencedEG}{n + m}
\]

\[
priorEG = priorEG + \alpha(parentEG - priorEG)
\]

n = 10; \ \alpha = .05; \ egss = .4; \ control \ rate \ of \ production \ learning
Activation \ threshold = 1, Noise = .4; \ controls \ location \ learning
Time to Select a Track

- Just Instruction Practice
- Location Learning
- Production Learning
- Production + Visual Learning
- Data

6 Min Trials
The Anatomy of Track Identification

End of the previous Unit Task
Anatomy of the Identification Unit Task

Time

Hook F6 F2 F9 F4 P F7 A F1

Selection Search Execute

Information gathering Initiate Classify Save
Anatomy of the Identification Unit Task

Time

Selection  Search  Execute

Information gathering  Initiate  Classify  Save

Hook  F6  F2  F9  F4  P  F7  A  F1
Anatomy of the Identification Unit Task

Time

Selection Search Execute

Information gathering Initiate Classify Save

Hook F6 F2 F9 F4 P F7 A F1
Anatomy of the Identification Unit Task

Selection  Search  Execute

Information gathering  Initiate  Classify  Save

Time

Hook  F6  F2  F9  F4  P  F7  A  F1
ID Track as Friendly, Non-Military

Select Non-Military

Hit the F9 Key

Find Save on the Menu

Select Save

Hit the F1 Key

Look to the Lower Left

Search to the Right for Save

23.321

23.621

24.139

24.289

23.571

25.309

23.471

24.909

24.289

24.139
The Last Interkey Press for Save

ID Track as Friendly, Non-Military

Select Non-Military

Select Save

338.586

338.786

338.836

Press Key

339.136

Press Key
51% of all eye fixation times are not to relevant regions
Anatomy of the Identification Unit Task

- **Information gathering**
- **Initiate**
- **Classify**
- **Save**

**Selection** → **Search** → **Execute**

**Time**

**Hook** → **F6** → **F2** → **F9** → **F4** → **P** → **F7** → **A** → **F1**

**Data: Scope** → **Data: Info** → **Data: F-Keys**

**Theory: Scope** → **Theory: Info** → **Theory: F-Keys**

Graph showing proportion fixation over 6-min trials.
Anatomy of the Identification Unit Task

- Selection
- Search
- Execute

Time:
- Hook
- F6
- F2
- F9
- F4
- P
- F7
- A
- F1

Data: Scope
Data: Info
Data: F-Keys

Theory: Scope
Theory: Info
Theory: F-Keys

Proportion Fixation

6-Min Trials
Conclusions

1. We seem to a viable mechanism for creating productions and learning from instruction. Why did it take so long?

2. There still are open issues as to what the best way to represent instructions and interpret them are and how to deal with flexibility
   (a) My solution involves default seriality
   (b) Niels solution involves explicitly representing ordering constraints

3. There are open issues about how to deal with control that come up in modeling tutoring.
   (a) Inserting override instructions
   (b) Evoking the instructions upon condition