Why study models of arithmetic?

Task is an example of a *cognitive skill* – acquired through conscious practice.
- driving (vs. walking)
- reading/writing (vs. understanding/speaking)

Model is an example of a *production system*.
- Often used to model cognitive skills.
- Useful in explaining how humans perform the task correctly by integrating many smaller subskills.
- Failure of individual subskills may help explain systematic failures in main skill.

Multi-column subtraction

How do skilled students perform this task?
What types of errors are made by learners?
- random errors versus systematic errors.
- factual (arithmetic) errors versus procedural errors.
- incorrect subskills versus failure to apply subskills.

Young and O’Shea (1981) hypothesized that many errors are caused by failing to apply a sub-component of the skill.
Basic architecture: Subtraction

Architecture is general production system, not specific to task:

- **Working memory**: holds current goals for task (multi-column subtraction) and subtasks (e.g. borrow).
- **Production memory**: holds production rules encoding when and how to perform subtasks (condition-action pairs) including arithmetic facts.
- **Match memory**: holds any production rules whose conditions are currently met.
- **Conflict resolution**: determines which rule in Match memory to fire.
- **Refractory memory**: keeps track of rules that have fired to prevent them firing again unless later reintroduced into Match memory.

Comparison to ACT-R

- **Working memory**: similar to ACT-R Goal module.
- **Production memory**: combines ACT-R production system and Declarative module.
- **Match memory**: Similar to ACT-R Retrieval buffer.
- **Conflict resolution**: Here, based on recency. ACT-R: based on subsymbolic activation levels.

Diagnosing Student Models

If teacher believes a student has a different model from their own (correct) one:

- make list (bug catalog) and match to it;
- reason about what student would have to believe in order to exhibit behavior indicating this.

**Student model**: representation of student’s current state of knowledge.

**Diagnosis**: process of inferring the student model.
Skilled examples

First, we need to understand the skill children are learning (maybe not the way all of you learned).

Problems with children’s work

- A: always subtract smaller digit from larger.
- B: always borrow.
- C: both A and B.
- D: subtracting larger number from smaller equals zero.
- E: borrowing makes 10.
- F: add instead of subtract.
- G, H: errors only with subtracting from zero.

Note that only patterns of errors distinguish G, H from A, D. Finding flaws in the underlying procedure (rather than specific errors) requires looking at multiple problems.

Young and O’Shea’s Model

Production rule model of multi-column subtraction:

- contains a fairly small number of simple production rules.
- children’s errors are modeled by deleting production rules from a model that works correctly.
- accounts for a large percentage of errors found in practice.
- supports hypothesis that many errors arise from forgetting a sub-component of the skill.
**A Simple Production Rule Model**

**Condition**
- S1: goal = process column & minuend greater than or equal to subtrahend
- S2: goal = process column & minuend less than subtrahend
- S3: goal = borrow

**Action**
- S1: Take absolute difference of minuend and subtrahend and write in the answer space
- S2: Push goal 'borrow' onto stack
- S3: Decrement next minuend by 1, add 10 to current minuend and delete the current goal

**Example**

S1 is the only applicable production, so it fires.

Now S1 is still the only applicable production! We need a fix...

* indicates current column

---

**A Revised Subtraction Model**

**Condition**
- S1: goal = subtract & all answer spaces empty
- S2: goal = process column & minuend greater than or equal to subtrahend
- S3: goal = process column & minuend less than subtrahend
- S4: goal = process column & answer space filled in
- S5: goal = borrow

**Action**
- S1: Place marker on rightmost column & push goal 'process column'
- S2: Take absolute difference of minuend and subtrahend and write in the answer space
- S3: Push goal 'borrow' onto stack
- S4: Move one column left
- S5: Decrement next minuend by 1, add 10 to current minuend and delete the current goal

**Example**

S1 is the only applicable production, so it fires. The marker is placed, the new goal put on the stack and S2 fires.

S2 and S4 both satisfy the conditions but recency rules out S2.
Example

<table>
<thead>
<tr>
<th>Subtract</th>
<th>Goal Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 9 minuend</td>
<td>-1 8 subtrahend</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
</tr>
</tbody>
</table>

S2's conditions are satisfied so it fires, then S4 will fire.

<table>
<thead>
<tr>
<th>Process Column</th>
<th>Goal Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 9 minuend</td>
<td>-1 8 subtrahend</td>
</tr>
<tr>
<td>3 1</td>
<td>*</td>
</tr>
</tbody>
</table>

Now no rules are satisfied so the system halts.

Revised Model Reconsidered

Condition
- S1: goal = subtract & all answer spaces empty
- S2: goal = process column & minuend greater than or equal to subtrahend
- S3: goal = process column & minuend less than subtrahend
- S4: goal = process column & answer space filled in
- S5: goal = borrow

Action
- Place marker on rightmost column & push goal 'process column'
- Take absolute difference of minuend and subtrahend and write in the answer space
- Push goal 'borrow' onto stack
- Move one column left
- Decrement next minuend by 1, add 10 to current minuend and delete the current goal

4 9
-1 8

OK

4 0 7
-1 0 8
not OK

Model needs to be revised further.

Young and O'Shea's rules

Condition
- Init: goal = subtract & all answer spaces empty
- Read: goal = process column & no M or S in working memory
- Compare: M and S in working memory
- FindDiff: M and S in working memory
- Borr2a: M < S
- BorrS1: goal = borrow
- BorrS2: goal = borrow
- AbsDiff: goal = find difference

Action
- Place marker on rightmost column & push goal 'process column'
- Read M and S
- Compare M and S
- Push goal 'find difference', push goal 'next column'
- Push goal 'borrow'
- Decrement next minuend by 1
- Add 10 to current minuend
- Take absolute difference between M and S as result
- Write result
- Move one column left
- Carry 1 and take X as result

4 8 7
-1 3 8

Model needs to be revised further.
Analysis of rules

- Why absolute difference?
  AbsDiff: goal = find difference → Take absolute difference between M and S as result
- What is the carry rule doing here?
  Carry: result is (1,X) → Carry 1 and take X as result

Faulty Models

Leaving out specific rules leads to many common errors.
- Compare: M and S in working memory → Compare M and S. If missing, take smaller from larger.
- BorrS1: goal = borrow → Decrement next minuend by 1. If missing, borrow freely, no payback.
  But not all:
  - Always borrow.
  - Zero errors.

Additional rules: borrowing

Replace
Borr2a: M < S → Push goal ‘borrow’
with one of these:
Borr2b: M > S → Push goal ‘borrow’
Borr1: M and S in working memory → Push goal ‘borrow’
- accounts for always borrow behavior.
- Young and O’Shea suggest teaching methods are to blame: students given only examples without borrowing, then only examples with borrowing. Never learn conditions for borrowing.

Additional rules: zeros

- Treated as additional production rules.
- Are these really procedural errors or arithmetic (factual) errors? Do students require more training in multi-column subtraction or arithmetic facts?
Arithmetic (multicolumn subtraction) as example of a cognitive skill;
- Using general architecture of a production system, subtraction can be modeled using specific production rules;
- Missing rules lead to degraded behavior similar to patterns of student errors;
- Diagnosis: inferring which skills (and subskills) students have mastered (or failed to master);

References
