Models of Arithmetic

LaLoCo, Fall 2013

Karl DeVries, Adrian Brasoveanu

[based on slides by Sharon Goldwater & Frank Keller]

Modeling Arithmetic Skill

- Motivation
- Architecture
- Diagnosing Student Models

A Production Rule Model

- A Basic Model
- A Revised Model
- Young and O'Shea's Model

Reading: Cooper (2002, Ch. 3)

Why study models of arithmetic?

An example of a *cognitive skill*: an ability learned through conscious practice. Others include:

- solving well-defined, knowledge-lean problems
- driving (vs. walking)
- reading/writing (vs. understanding/speaking)

Focus on *multi-column subtraction*; Cooper also covers addition. Both models illustrate how

- cognitive skills can be modeled using a *production system*.
- humans perform a task correctly by integrating many smaller sub-skills;
- failure of individual sub-skills may help explain systematic failures in main skill.

Multi-column subtraction

How do skilled students perform this task?

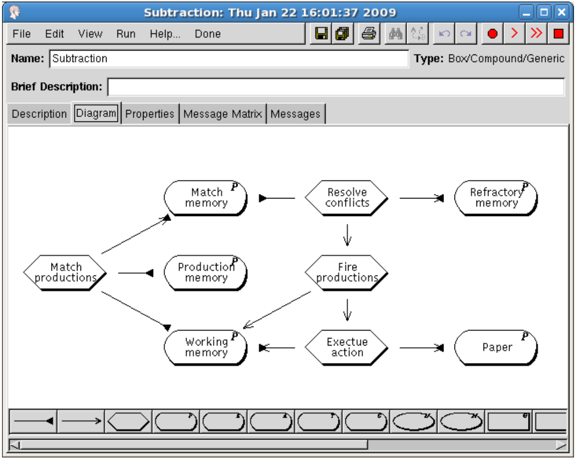
What types of errors are made by learners?

- random errors or systematic errors?
- factual (arithmetic) errors or procedural errors?
- incorrect sub-skills or failure to apply sub-skills?

Young and O'Shea (1981) hypothesized that many errors are caused by failing to apply a sub-skill.

Basic architecture: General production system

Architecture is a general *production system*, not specific to this task:



Basic architecture: General production system

- **Working memory**: holds current goals for task (here, multi-column subtraction) and subtasks (e.g., borrow).
- **Production memory**: holds production (ie, condition-action) rules encoding *when* and *how* to perform subtasks. Also stores relevant facts (here, arithmetic facts).
- **Match memory**: holds production rules whose conditions are currently met.
- **Resolve conflicts**: if $>$1 rule in **Match memory**, determines which to fire.
- **Refractory memory**: keeps track of rules that have fired, in order to prevent the same rule from firing multiple times (unless re-introduced into **Match memory**).

Comparison to ACT-R

Recall that ACT-R is also a general production system. Not surprisingly, the architectures of the two systems are similar.

- **Working memory**: similar to ACT-R Goal module.
- **Production memory**: combines ACT-R Central 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:

- assemble a bug catalog;
- 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.

Examples of children's work

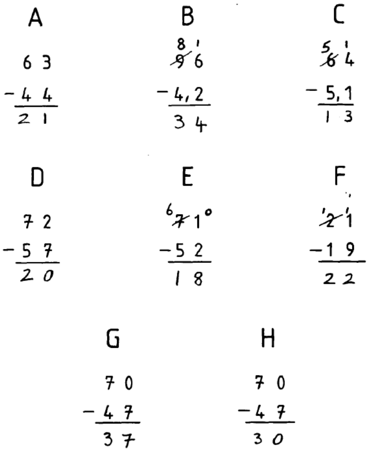


Figure 1. Examples of subtraction errors.

Figure from Young and O'Shea (1981)

Problems with children's work

Terminology:

YYY	*minuend*
XXX	*subtrahend*
ZZZ	*difference*

Errors:

- 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 (rather than 10+*minuend*).
- 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		**Action**
S1: (goal = process column) & (*minuend* $\geq$ *subtrahend*)	$\longrightarrow$	Take absolute difference of *minuend* and *subtrahend* and write in the answer space
S2: (goal = process column) & (*minuend* $<$ *subtrahend*)	$\longrightarrow$	Push goal 'borrow' onto stack
S3: (goal = borrow)	$\longrightarrow$	Decrement next minuend by 1, add 10 to current *minuend* and delete the current goal

Example

process column	4	9	minuend
goal stack	–1	8	*subtrahend*
		*	

S1 is the only applicable production, so it fires.

process column	4	9	minuend
goal stack	–1	8	*subtrahend*
		1	
		*	

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

* indicates current column

A Revised Subtraction Model

Condition		Action
S1: (goal = subtract) & all answer spaces empty	$\longrightarrow$	Place marker on rightmost column & push ‘process column’ onto goal stack
S2: (goal = process column) & (*minuend* $\geq$ *subtrahend*)	$\longrightarrow$	Take absolute difference of *minuend* and *subtrahend* and write in the answer space
S3: (goal = process column) & (*minuend* $<$ *subtrahend*)	$\longrightarrow$	Push goal ’borrow’ onto stack
S4: (goal = process column) & answer space filled in	$\longrightarrow$	Move one column left
S5: (goal = borrow)	$\longrightarrow$	Decrement next *minuend* by 1, add 10 to current *minuend* and delete the current goal

Example 1

subtract

goal stack

4	9	*minuend*
–1	8	*subtrahend*
	*	

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

process column subtract

goal stack

4	9	*minuend*
–1	8	*subtrahend*
	1	
	*	

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

Example 1

goal stack
subtract

4	9	*minuend*
–1	8	*subtrahend*
	1	
*		

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

goal stack
process column
subtract

	4	9	*minuend*
	–1	8	*subtrahend*
	3	1	
*			

Now no rules are satisfied so the system halts.

Example 2

Condition		**Action**
S1: (goal = subtract) & all answer spaces empty	$\longrightarrow$	Place marker on rightmost column & push ‘process column’ onto goal stack
S2: (goal = process column) & (*minuend* $\geq$ *subtrahend*)	$\longrightarrow$	Take absolute difference of *minuend* and *subtrahend* and write in the answer space
S3: (goal = process column) & (*minuend* $<$ *subtrahend*)	$\longrightarrow$	Push goal ’borrow’ onto stack
S4: (goal = process column) & answer space filled in	$\longrightarrow$	Move one column left
S5: (goal = borrow)	$\longrightarrow$	Decrement next *minuend* by 1, add 10 to current *minuend*, delete current goal

$$\begin{array}{rr} 4 & 8 \\ -1 & 9 \\ \hline \end{array} \quad \text{OK}$$

$$\begin{array}{rrr} 4 & 0 & 7 \\ -1 & 0 & 8 \\ \hline \end{array} \quad \text{not OK:}$$

(How to borrow from 0?)

Young and O'Shea's Model

Production rule model of multi-column subtraction:

- contains a fairly small number of simple production rules.
- models children's errors by deleting 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-skill.

Young and O'Shea stress that rules do not form a structurally delimited module: If during subtraction, circumstances are appropriate for triggering other rules, they will fire.

Young and O'Shea's Production Rules

Condition		Action
`Init`: goal = subtract & all answer spaces empty	$\longrightarrow$	Place marker on rightmost column & push goal 'process column'
`Read`: goal = process column & no M or S in working memory	$\longrightarrow$	Read M and S
`Compare`: M and S in working memory	$\longrightarrow$	Compare M and S
`FindDiff`: M and S in working memory	$\longrightarrow$	push goal 'find difference', push goal 'next column'
`Borr2a`: $M < S$	$\longrightarrow$	Push goal 'borrow'
`BorrS1`: goal = borrow	$\longrightarrow$	Decrement next *minuend* by 1
`BorrS2`: goal = borrow	$\longrightarrow$	Add 10 to current *minuend*
`AbsDiff`: goal = find difference	$\longrightarrow$	Take absolute difference between M and S as result
`Write`: result in working memory	$\longrightarrow$	Write result
`Next`: goal = process column & answer space filled in	$\longrightarrow$	Move one column left
`Carry`: result is (1,X)	$\longrightarrow$	Carry 1 and take X as result

Faulty Models: Missing rules

Leaving out specific rules leads to many common errors.

- `Compare`: M and S in working memory $\longrightarrow$ Compare M and S.

  If missing, *take smaller from larger*.

- `BorrS1`: goal = borrow $\longrightarrow$ Decrement next *minuend* by 1.

  If missing, *borrow freely, no payback*.

But not all: Additional errors may come from faulty rules

- Always borrow.
- Zero errors.
- ...

Additional faulty rules: borrowing

Replace

`Borr2a`: M $<$ S $\longrightarrow$ Push goal 'borrow'

with one of these:

`Borr2b`: M $>$ S $\longrightarrow$ Push goal 'borrow'

`Borr1`: M and S in working memory $\longrightarrow$ 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 faulty rules: zeros

Condition		**Action**
`Nmin00`: M=N, S=0	$\longrightarrow$	result is 0
`0minNN`: M=0, S=N	$\longrightarrow$	result is N
`0minN0`: M=0, S=N	$\longrightarrow$	result is 0
`NminNN`: M=N, S=N	$\longrightarrow$	result is N

- 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?

Summary

- 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 I

Cooper, Richard P. (2002). *Modelling High-Level Cognitive Processes*. Mahwah, NJ: Lawrence Erlbaum Associates.
Young, R. M. and T. O'Shea (1981). "Errors in Children's Subtraction". In: 5.2, pp. 153–177.