ISE

ISE-TR-91-89

The Subtraction Problem Generating

by

I. Shevchenko and K. Nakayama

March 11, 1991

INSTITUTE
OF
INFORMATION SCIENCES AND ELECTRONICS
UNIVERSITY OF TSUKUBA

The Subtraction Problem Generating

I. Shevchenko[1] and K. Nakayama[2]

[1] Department of Mathematics, Far-Eastern State University, Vladivostok,690600 U.S.S.R. (this work has been completely prepared while this author was visiting the University of Tsukuba).

[2] Institute of Information Sciences and Electronics, University of Tsukuba, Tsukuba,305 Japan.

Abstract

We implemented in C-prolog the prototypes of two programs which produce the multi-column subtraction problems specific to a set of conditions. The first program makes use of a production system model for performing written subtraction. The flow of randomly generated problems with some "static" properties is run through the model. The process of interpretation is traced, and the program picks only the problems which possess specified "dynamic" properties. The second program is given a block structure type and a set of the subtraction facts. The block structure type is a specification of some essential processing properties of the problem. Abstract interpretation of the subtraction procedure adds relations between variables which represent initial and actual column operands and results. Then all possible patterns with instances of the required subtraction facts are produced. A concrete problem is generated by a random consistent instantiation of the remained variables in a chosen pattern. We also analysed types of the tests being used in elementary school in Japan during the first year of teaching multi-column subtraction.

Introduction

Although carrying out multi-column subtraction in the different bases is among initial arithmetic skills, this is rather complicated activity possessed such characteristics like the form and the levels of generality, details, autonomy and drilling. There exist material, perceiving, oral and intellectual forms. The level of generality describes the portion of acquired combinations of subprocedures that constitute the activity. The level of details is defined by the number and size of components included into performing the subprocedures. The level of autonomy reflects the needed help. The level of drilling is measured by the speed of carrying out. First the activity must be taken step-by-step to the required form and levels of generality, details and autonomy, and only then can be intensively drilled[1]. Instruction design to convert carrying out subtraction from external (material) form through the watching-when-performing and speaking-when-performing forms into internal (intellectual) form is difficult enough from psychological and pedagogical as well as from information processing viewpoints. Much research on computers in teaching subtraction is concerned with detacting and remeditating buggy subprocedures (see, e.g. [2],[3]). For example, in [2] 61 variants of possible subtraction subskills from knowing how carry out any problem to the lack of any skill are described. It should be more productive to design the instructions and tests in order to prevent mislearning than to make every effort for remeditation[4].

The goal of this research is to develop the programs which can be employed by instruction/test designers and computerized tutors for analysing and generating subtraction problems of the different generality level. Two ways of approaching the problem are examined. The first is based on the imitation of the writing subtraction procedure by a production system model. The other makes use of abstract interpretation of the procedure(see,e.g. [5]). We also describe the type analysis of the subtraction problem tests being used in elementary school in Japan during the first year of teaching multi-column subtraction[6].

The imitational approach

Learning to carry out subtraction in the different bases means not only acquiring specific, but also some logic and general knowledge, like informal evaluation of compound propositions using logic operations or acting according to a complete and consistent set of rules. All these skills can be mastered together, but we concentrate here mainly on on the subtraction rules and facts tables.

The decomposition method of multi-column subtraction[3] is represented as a production system (Fig. 1). The written actions occur at the special place in the working memory called the "blackboard". To generate the problems of the different generality levels required some subtraction facts, the program must merely generate initial states of the blackboard that provide firing a certain sequence of production rules. The tracer is included to follow the interpretation process. For example, if the initial state of the blackboard corresponds to the Sample

$$\begin{array}{lr} & 1\ 0\ 1\ 7_{10} \\ \text{(What is)} & -\ 6\ 5\ 8_{10}\ \ \text{(?)} \\ \hline \end{array}$$

then the final state corresponds to the answer [1]

```
          1
 0    9   0   1
 1.   0.  1.  7₁₀
 -    6   5   8₁₀
-------------------
      3   5   9₁₀
```

and the sequence of visible rules [2]

```
trial-to-perform-column-subtraction-when-less(...),
perform-column-subtraction-when-equal-or-greater(...),
minus(10,17,8,9),
trial-to-perform-column-subtraction-when-less(...),
zero-skipping(...),
perform-column-subtraction-when-equal-or-greater(...),
minus(10,10,5,5),
perform-column-subtraction-when-equal-or-greater(...),
minus(10,9,6,3),
perform-column-subtraction-when-equal-or-greater(...),
minus(10,0,0,0),
```

is fired.

Fig.2 shows the information flow of the program based on the production system model and generate-and-test technique[7]. Such elements of the set of static facts as lengths of operands are employed for creating a frame of the subtraction problem. Then this frame is filled in with optional static facts like particular column digits. A random-digit generator fills in remained positions. The obtained problem is run through the production system model to check meeting the set of dynamic properties.

For example, to generate the subtraction problem similar to the Sample in the base 10, we should include into the set of static facts such facts as

```
minuend-length(...,4),
subtrahend-length(...,3),
minuend-digit-at-first-column(...,1),
```

and into the set of dynamic facts such facts as

```
result-length(...,3),
number-of-trials-to-perform-when-less(...,2),
number-of-zero-skipping(...,1).
```

In this case the output sequence contains such problems as

[1] The dot "." stands for the striking out.

[2] We abbrivate most argument lists and thier parts with ..., since printing out represntations of the blackboard and other structures takes too much space.

```
              1  5  0  5₁₀
(What is )    -  7  2  9₁₀   (?)
```

and

```
              1  0  9  0₁₀
(What is )    -  8  9  6₁₀   (?)
```

The blind trial-and-error technique leads to inefficiency of the program because the generated flow of the problems with static properties is not uniformly distributed over dynamic properties, so that the program usually needs a lot of trials to pick the problem. Besides that, the program can go into the infinite futile cycle in case the sets of static and dynamic properties are inconsistent.To improve the program characteristics, instead of random generating the next candidate we could introduce the "varying-an-existing-problem" heuristics, like increamenting/decreamenting some digits or switching/adding/deleting columns [2]. However our ultimate goal is to avoid testing entirely.

An abstract specification of the subtraction problems

Let b be the base, $V_{k,0} = v_k \ldots v_1 v_0$ be a representation of the number $v_k b^k + \ldots v_1 b^1 + v_0 b^0$ and $V_{n,m}$ be a subvector of $V_{k,0}$, $0 \leq m < n \leq k$. The decomposition method of subtraction algorithm can be described as two rewrite rules:

1.

$$\begin{array}{cccc} M_{k,i+1} & m_i & M_{i-1,0} & \\ -S_{k,i+1} & s_i & S_{i-1,0} & \Rightarrow \\ \hline & & D_{i-1,0} & \end{array} \quad \begin{array}{cccc} M_{k,i+1} & m_i & M_{i-1,0} \\ -S_{k,i+1} & s_i & S_{i-1,0} \\ \hline & d_i & D_{i-1,0} \end{array}$$

where $m_i \geq s_i$, $d_i = m_i - s_i$.

2.

$$\begin{array}{ccccc} M_{k,i+1} & m_j & 0_{j-1,i+1} & m_i & M_{i-1,0} \\ -S_{k,i+1} & s_j & S_{j-1,i+1} & s_i & S_{i-1,0} \\ \hline & & & & D_{i-1,0} \end{array} \Rightarrow \begin{array}{ccccc} M_{k,i+1} & m'_j & b'_{j-1,i+1} & m'_i & M_{i-1,0} \\ -S_{k,i+1} & s_j & S_{j-1,i+1} & s_i & S_{i-1,0} \\ \hline & & & & D_{i-1,0} \end{array}$$

where $m_i < s_i$, $m_j > 0$, $m'_j = m_j - 1$, $m'_i = m_i + b$, $b' = b - 1$; $0_{j-1,i}$ and $b'_{j-1,i}$ are constant subvectors with zero and base-minus-one components respectively.
Therefore, most essential processing properties of the problem can be described by the following five types of the structure blocks

$$B_<, B_<^{0(j)}, B_=, B_=^{0(j)}, B_>,$$

where

$$B_R = B_R(m,s,d) = \begin{bmatrix} m \\ s \\ d \end{bmatrix},$$

$$B_R^{0(j)} = B_R^{0(j)}(m,s,d;m_1,s_1,d_1;\ldots;m_j,s_j,d_j) = \begin{bmatrix} 0 & \ldots & 0 & m \\ s_j & \ldots & s_1 & s \\ d_j & \ldots & d_1 & d \end{bmatrix},$$

and $mRs, m > 0, m_1 = 0, \ldots, m_j = 0, s_1 \geq 0, \ldots, s_j \geq 0$. Splitting into the blocks of a certain particular subtraction problem is carried out in order from right to left column, so that unique block structure type is produced. For instance, the Sample has the block structure of the $B_> B_<^{0(1)} B_<$ type. Any problem of the same type fires the same sequence of the production rule names (see Section 1).

The block structure type can also be employed for describing the order of presentation of the different generality level problems. For instance, Table 1 gives a description in time and in quantity of the subtraction tests being used in elementary school in Japan during the first year of teaching subtruction procedure (see [6]).

Abstract interpretation approach

The block structure type is a description of the relations between variables which represent the column digits. During carring out subtraction some borrowings can occur and actual column operands can differ from the initial digits. Abstract interpretation can be divided into two stages: insertion of the borrowing marks and generation of of the column operans-result relations.

The marked type is produced according to the following transformation rules applied to the type in order from left to right

$$\begin{array}{llllllll}
B_< & & \Rightarrow & \beta & B_<, & & & \\
B_< & \beta & \Rightarrow & \beta & B_< & \beta, & & \\
B_> & & \Rightarrow & & B_>, & & & \\
B_> & \beta & \Rightarrow & & B_> & \beta, & & \\
B_= & & \Rightarrow & & B_=, & & & \\
B_= & \beta & \Rightarrow & \beta & B_= & \beta, & & \\
B_<^{0(j)} & & \Rightarrow & \beta & B_<^{0(j)}, & & & \\
B_<^{0(j)} & \beta & \Rightarrow & \beta & B_<^{0(j)} & \beta, & & \\
B_=^{0(j)} & & \Rightarrow^* & \beta & B_<^{0(j-1)} & & B_=, & \\
B_=^{0(j)} & & \Rightarrow^* & \beta & B_<^{0(j-2)} & & B_= & B_=, \\
\ldots & & & \ldots & \ldots & & \ldots & \ldots \\
B_<^{0(j)} & & \Rightarrow^* & & & & B_= & B_= \quad \ldots \quad B_=, \\
B_=^{0(j)} & \beta & \Rightarrow & \beta & B_=^{0(j)} & \beta, & &
\end{array}$$

where β stands for the borrowing[3]. For instance, the marked type $B_> \beta B_<^{(0(1)} \beta B_<$ corresponds to the type $B_> B_<^{0(1)} B_<$.

[3] The asterisk marks alternative rules. The program merely makes use of the first of them.

Using marked type actual relations between column operands and results can be easily deduced. For example,

$$
\begin{array}{llllllllll}
\beta & B_{<} & & \rightarrow & \{ & & m<s, & m'=m+b, & d=m'-s & \}, \\
\beta & B_{<}^{0(j)} & & \rightarrow & \{ & & m<s, & m'=m+b, & d=m'-s, & \\
 & & & & & m_1=0, & s_1\geq 0, & m_1'=m_1+b, & d_1=m_1'-s_1, & \\
 & & & & & \cdots & \cdots & \cdots & \cdots & \\
 & & & & & m_j=0, & s_j\geq 0, & m_j'=m_j+b, & d_j=m_j'-s_j & \}, \\
\beta & B_{<} & \beta & \rightarrow & \{ & & m<s, & m'=m+b-1, & d=m'-s & \}, \\
\beta & B_{<}^{0(j)} & \beta & \rightarrow & \{ & & m<s, & m'=m+b-1, & d=m'-s, & \\
 & & & & & m_1=0, & s_1\geq 0, & m_1'=m_1+b, & d_1=m_1'-s_1, & \\
 & & & & & \cdots & \cdots & \cdots & \cdots & \\
 & & & & & m_j=0, & s_j\geq 0, & m_j'=m_j+b, & d_j=m_j'-s_j & \}, \\
 & B_{>} & \beta & \rightarrow & \{ & & m>s, & m'=m-1, & d=m'-s & \}, \\
\beta & B_{=} & \beta & \rightarrow & \{ & & m=s, & m'=m+b-1, & d=m'-s & \}.
\end{array}
$$

Thus the result of these successive transformations is a set of the relations. To generate the problem of a particular type required some subtraction facts, first the program creates pattern(s) by instantiating variables in the relations according to those facts and then, using a random-digit generator, instantiates remained variables (see Fig. 3).

For example,the problem frame

$$
\begin{array}{cccc}
m_4 & m_3 & m_2 & m_1 \\
-s_4 & s_3 & s_2 & s_1 \\
\hline
d_4 & d_3 & d_2 & d_1
\end{array}
$$

where $m_1 > s_1, m_2 < s_2, m_3 = 0, m_3 < s_3, m_4 > s_4$, corresponds to the block structure type $B_{>}B_{<}^{0(1)}B_{<}$. If there are no required facts, then the result of abstract interpretation is the following set of the relations in the base b

$$
\begin{array}{llll}
\{ & m_1<s_1, & m_1'=m_1+b, & d_1=m_1'-s_1, \\
 & m_2<s_2, & m_2'=m_2+b-1, & d_1=m_2'-s_2, \\
m_3=0, & m_3<s_3, & m_3'=b-1 & d_3=m_3'-s_3, \\
 & m_4>s_4, & m_4'=m_4-1, & d_4=m_4'-s_4 \quad \},
\end{array}
$$

and in the base 10 random instantiation leads to the subtraction problems like

$$
\begin{array}{lcccc}
 & 4 & 0 & 1 & 1_{10} \\
\text{(What is)} & -2 & 9 & 5 & 7_{10} \quad \text{(?)} \\
\hline
\end{array}
$$

If the only required fact is

$$0 - 0 = 0(10),$$

then the output flow contains the problems similar to the Sample, like

$$
\begin{array}{lcccc}
 & 1 & 0 & 2 & 5_{10} \\
\text{(What is)} & - & 3 & 7 & 6_{10} \quad \text{(?)} \\
\hline
\end{array}
$$

In the case when one more required fact,

$$17 - 9 = 8,$$

is added, the program provides a choice between two patterns with instantiated digits of the first or second column. In the former case the program generates the problems like

```
             1  0  3  7₁₀
(What is )   -  4  5  9₁₀   (?)
```

and in the latter case like

```
             1  0  8  2₁₀
(What is )   -  8  9  9₁₀   (?)
```

Conclusion

In teaching the multi-column subtraction procedure in addition to psychological and pedagogical aspects there are some information processing problems related to the generality level of the skill. To prevent having students solve the subtraction problems which are beyond mastered skills, the instruction/test designers or the computerized tutors need a generator producing the problems with merely specified properties. We have employed imitational and abstract interpretation techniques for making the prototypes of two programs that analyse and generate the subtraction problems in the different bases. Using the type specifications we have also given a description of the tests being used in elementary school in Japan during the first year of teaching subtraction procedure. The procedural domain of multi-column subtraction was chosen only as an example of mastered elementary arithmetic skill.

Acknowledgements

We would like to thank John Nesbit for helpful discussions at the preliminary stage of the development of this work. Further thanks are due to Nobuhito Yamamoto who stimulated our interest in symbolic computation methods and who provided facilities to carry out the project.

Bibliography

[1] Talyzyna, N.F., *Formation of Younger Pupil's Cognitive Activity.* Moscow: Prosveshchenie,(1988). (In Russian.)

[2] Burton, R.R., Diagnosing Bugs in a Simple Procedural Skill. In D. H. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems.* New York: Academic,(1982).

[3] Young, R.M., O'Shea, T., Errors in Children's Subtraction. *Cognitive Science, 5*,(1981),pp. 152-177.

[4] VanLehn, K., Two pseudo-students: Application of Machine Learning to Formative Evaluation. In *Proceedings of the International Conference on Advanced Research on Computers in Education.* Tokyo,(1990), pp. 181-190.

[5] Evertsz, R., Refining the Student's Procedural Knowledge Through Absract Interpretation. In D. Bierman, J. Breuker & J. Sandberg (Eds.), *Artificial Intelligence and Education.* Amsterdam: IOS,(1990), pp. 102-106.

[6] Fujiwara, Y., (Ed.), *The Training Papers. Elementary School. Second Grade. Arithmetic. The A Course.* April - March. Tokyo: Kyoikusha. (In Japanese.)

[7] Sterling, L., Shapiro, E. Y., *The Art of Prolog.* Cambridge, MA: MIT Press,(1986).

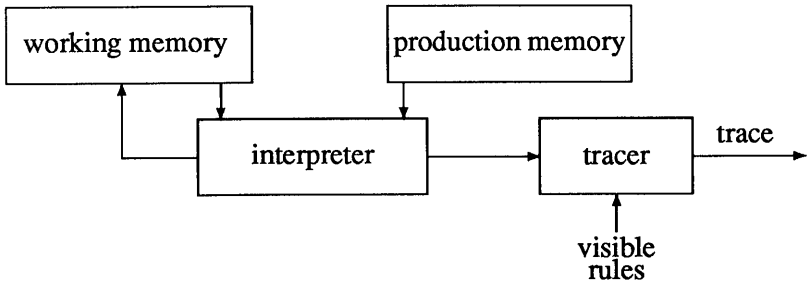

Figure 1: The production system model.

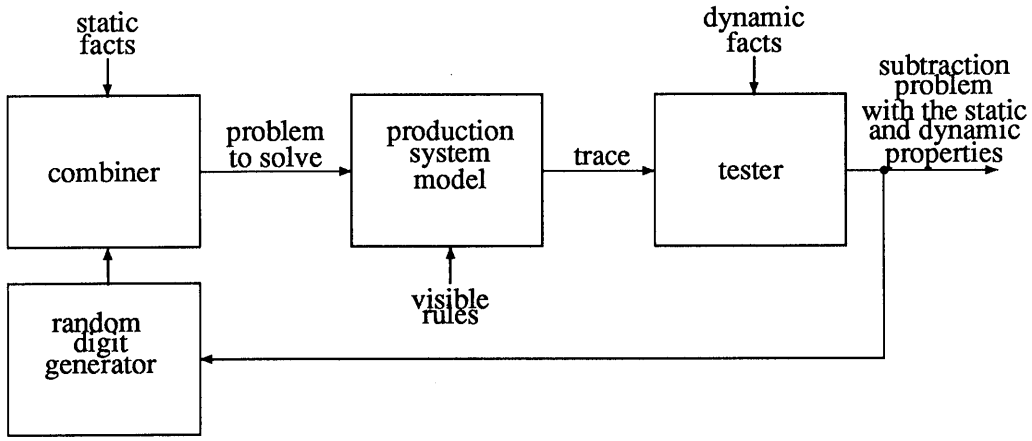

Figure 2: Information flow of the program based on imitational approach.

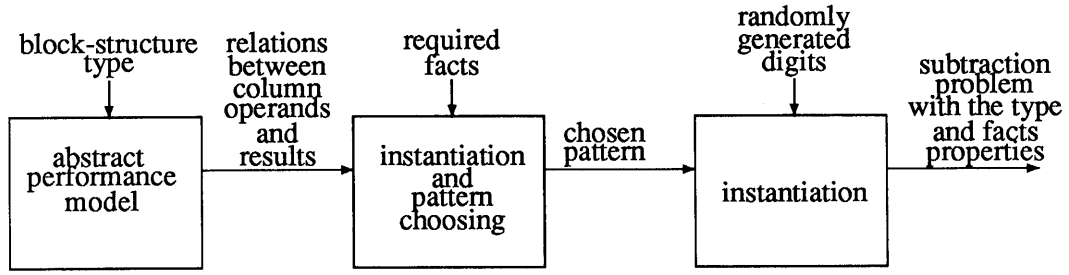

Figure 3: Information flow of the program based on abstract performance approach.

Table 1. Type analysis of the subtraction tests from [6]

Block-structure types	Month										Totals
	4	5	7	8	9	10	12	1	2	3	
$B_>B_>, B_>B_=, B_=B_>$	109	101	8	19		8		5	3		253
$B_>B_<$		180	20	27		14		18	6	1	266
$B_>B_>B_>, B_>B_>B_=,$ $B_>B_=B_=, B_>B_=B_>,$ $B_=B_=B_>, B_=B_>B_=$					48	4	4		2	4	62
$B_>B_>B_<, B_=B_>B_<,$ $B_>B_<B_>, B_>B_<B_=$					143	15	13		4	12	187
$B_>B_<B_<, B_>B_=B_<$						79	8		3	9	99
$B_>B_<^{0(1)}$						105	3		1	4	113
$B_>B_>B_=B_>, B_>B_>B_>B_=,$ $B_>B_=B_>B_=$										3	3
$B_>B_<B_>B_>, B_>B_<B_=B_>,$ $B_>B_<B_>B_=, B_>B_>B_>B_<,$ $B_>B_<B_=B_=$									25	14	39
$B_>B_<B_>B_<$									6		6
$B_>B_<B_<B_=, B_>B_>B_=B_<,$ $B_>B_>B_<B_<, B_>B_=B_<B_>,$ $B_>B_<B_<B_>$									4	6	10
$B_>B_<B_<B_<, B_>B_=B_<B_<,$ $B_>B_<B_=B_<, B_>B_=B_=B_<$									18	6	24
$B_>B_<^{0(1)}B_<, B_>B_<B_<^{0(1)},$ $B_>B_<^{0(1)}B_>$									9	1	10
$B_>B_<^{0(2)}$									42	7	49
Totals	109	281	28	46	191	225	28	23	123	67	1121

INSTITUTE OF INFORMATION SCIENCES AND ELECTRONICS
UNIVERSITY OF TSUKUBA
TSUKUBA-SHI, IBARAKI 305 JAPAN

REPORT DOCUMENTATION PAGE

REPORT NUMBER
ISE-TR-91-89

TITLE

The Subtraction Problem Generating

AUTHOR(S)

I. Shevchenko K. Nakayama

REPORT DATE
March 11 , 1991

NUMBER OF PAGES
9 (the text)

MAIN CATEGORY
Computers and Education

CR CATEGORIES

KEY WORDS
problem generation, subtraction
production system model, abstract interpretation

ABSTRACT

We implemented in C-prolog the prototypes of two programs which produce the multi-column subtraction problems specific to a set of conditions. The first program makes use of a production system model for performing written subtraction. The flow of randomly generated problems with some "static" properties is run through the model. The process of interpretation is traced, and the program picks only the problems which possess specified "dynamic" properties. The second program is given a block structure type and a set of the subtraction facts. The block structure type is a specification of some essential processing properties of the problem. Abstract interpretation of the subtraction procedure adds relations between variables which represent initial and actual column operands and results. Then all possible patterns with instances of the required subtraction facts are produced. A concrete problem is generated by a random consistent instantiation of the remained variables in a chosen pattern. We also analysed types of the tests being used in elementary school in Japan during the first year of teaching multi-column subtraction.

SUPPLEMENTARY NOTES