Appendix A Trace of the template type 10 algorithm on an example problem

Figure 7 shows the trace of the template type 10 algorithm on an example math problem.

Cynthia works at a grocery store. She knows that for every 2 cans of tomato soup she sells , she will sell 4 cans of chili beans. This week she ordered a total of 12 cans. How many cans of chili beans did she order?

What do she order? –> 12 cans
What does She know? –> for every 2 cans of tomato soup she sells , she will sell 4 cans of chili beans $\rightarrow$ **(2, 4) pair occurrence.**
Where does Cynthia work? –> a grocery store $\rightarrow$ **pruned**.
What did she order This week? –> 12 cans
Who does work at a grocery store? –> cynthia $\rightarrow$ **pruned**.
How many cans did she order This week? –> 12 $\rightarrow$ $n3$ **becomes** 12.
When did she order a total of 12 cans? –> this week
When will she sell 4 cans of chili beans? –> for every 2 cans of tomato soup she sells , she will sell 4 cans of chili beans . this week $\rightarrow$ **(2, 4) pair occurrence.**
Does she sell every 2 cans of tomato soup? –> she knows that for every 2 cans of tomato soup she sells , she will sell 4 cans of chili beans $\rightarrow$ **(2, 4) pair occurrence.**
Do she order How many cans of chili beans? –> 12 cans
How many cans of tomato soup does she sell? –> 2
Did she order a total of 12 cans This week? –> this week she ordered a total of 12 cans
What will she sell for every 2 cans of tomato soup she sells? –> 4 cans of chili beans
How many cans . How many cans of chili beans did she order does She know? –> 12 cans
Will she sell 4 cans of chili beans for every 2 cans of tomato soup she sells? –> for every 2 cans of tomato soup she sells , she will sell 4 cans of chili beans $\rightarrow$ **(2, 4) pair occurrence.**
How many cans of chili beans will she sell for every 2 cans of tomato soup she sells? –> 4
How many cans of tomato soup she sells will sell for every 4 cans of chili beans for every 2 cans of tomato soup she sells? –> 12 $\rightarrow$ **(2, 4) pair occurrence.**
How many cans of chili beans . This week she ordered a total of 12 cans . How many cans of chili beans did she order does She know? –> 4
Does She know that for every 2 cans of tomato soup she sells, she will sell 4 cans of chili beans . This week she ordered a total of 12 cans . How many cans of chili beans did she order? –> she $\rightarrow$ **pruned.**

==> Template: "n1*x = n2*y", "x+y = n3"
==> Solution Equation: 4.0*tomato=2.0*chili and tomato+chili=12.0

$\rightarrow$ **The most frequent number pair is** $(2, 4)$ **with the total count** 5**.**
$\rightarrow$ **Number** 2 **appears before the number** 4 **in the problem text. Thus,** $n2$ **is** 2 **and** $n1$ **is** 4**.**
$\rightarrow$ **The assigned** n **values** $(4, 2, 12)$ **match with those in the solution equation.**

Figure 7. An example math problem solution using the rules from Figure 5.

Given the question-answer pairs generated out of the problem text, the first part of processing prunes some of the question-answer pairs by the first while loop filters. The co-occurrence of numbers (2 and 4 in this case) in the generated answers is saved as number tuples in a dictionary. Number 12 is assigned into $n3$, being the sole number as the response to a "How many" question. Last, the most frequent tuple is $(2, 4)$ and the number 2 appears before 4 in the problem text. Thus, $n2$ is 2 and $n1$ is 4. All the assigned n values match with those in the solution equation.

Appendix B Pseudocode for the equation template 15

We also provide another pseudocode for a more complicated equation template (Template Type 15) in Figure 8. The code starts with a similar pruning step. Afterward, a dictionary is created to hold the frequencies of potential number pairs in the answer texts. This time, we exploit two number pairs from the dictionary, the most and second most frequent, to make assignments to number slots. Firstly, we refer to the former to fill in the number slots $n1$ and $n2$. But, if the math problem mentions percentages, we adjust the numbers by multiplying them by 0.01. Later, the latter is placed into the slots $n3$ and $n4$ by using word similarities. Specifically, we consider the similarity of succeeding words in the problem text to those numbers to decide placement. If all the number slots are complete, the equation template is correctly instantiated, thus is correctly solved.

```
Data: a math problem P, $S = \{(q_1, a_1), ..., (q_k, a_k)\}$, ["n1*x + n2*y = n3", "x + y = n4"]
while for all (q, a) ∈ S do
    if q contains no number or number related words then
        set this (q, a) pair to be ignored;
    end
    if q starts with the word "Where" Or "When" then
        set this (q, a) pair to be ignored;
    end
    if q contains more than 40 words then
        set this (q, a) pair to be ignored;
    end
end
create a dictionary D to hold the appearance count of potential number pairs;
while for all (q, a) ∈ S such that its q is not set to be ignored do
    if q and a have at least 2 number values in total then
        find all potential number pairs for each number in the q and a group;
        increase the count of each number pair appearance in the dictionary D;
    end
end
assign (m1, m2) as the number pair with the max appearance count;
assign (m3, m4) as the number pair with the second highest appearance count;
if P contains the words "percent" Or "%" Or "cents" then
    m1 ← 0.01 * m1;
    m2 ← 0.01 * m2;
end
if m1 appears before m2 in P then
    n1 slot ← the number in m1;
    n2 slot ← the number in m2;
else
    n2 slot ← the number in m1;
    n1 slot ← the number in m2;
end
w1 ← the word following m1 in P ;
w2 ← the word following m2 in P ;
w3 ← the word following m3 in P ;
w4 ← the word following m4 in P ;
if w3 is a similar word compared to one of the words in the (w1, w2) pair then
    n3 slot ← the number in m3;
    n4 slot ← the number in m4;
end
if w4 is a similar word compared to one of the words in the (w1, w2) pair then
    n3 slot ← the number in m4;
    n4 slot ← the number in m3;
end
if n1 Or n2 Or n3 Or n4 is empty then
    return a flag that shows this math problem as unsolvable;
else
    return n1 And n2 And n3 And n4
end
```

Figure 8. Solution algorithm for the template type 15 problems.

Appendix C The Logic Behind the Rule Sets

There are 24 rule sets (i.e. number assignment algorithms) in total. Most of the rule sets share a similar logic of number assignment steps that can be called sub-rules. In Figures 9, 10, 11 and 12; one can see some examples of these sub-rules.

In Figure 9, the most common sub-rule can be seen. This sub-rule exists in every rule set. The main purpose of this sub-rule is to eliminate generated questions and answers that potentially do not contain any useful information. This sub-rule increases efficiency as well.

The sub-rule shown in Figure 10 is used to count the number of significant number groups inside the generated questions and answers. Generally, the most frequent number group can be instrumental while assigning the numbers to empty slots in the other types of sub-rules. This particular example has been extracted from the rule-set of the type 15 problems. However, this sub-rule has many variations in the different rule sets, depending on their specific requirements.

The sub-rule, shown in Figure 11 assigns the number pair with the max occurrence count to the corresponding empty slots in the correct order. This order is usually determined by checking which number appears before the other one in the original problem P. This particular example has been extracted from the rule-set of the type 10 problems. However, this sub-rule has some different variations in the other rule sets depending on their specific requirements.

The sub-rule, shown in Figure 12 assigns a secondary number pair after the first number pair has been assigned. The second pair can be assigned to the remaining empty slots by comparing the similarity of the succeeding words of each number in the original problem P. This particular example has been extracted from the rule-set of the type 15 problems. However, this sub-rule has many variations in the different rule sets depending on their specific requirements.

As seen from these example sub-rules, most rule sets can be formed by variations of these sub-rules depending on the requirements of a problem type. For example, a variation of the sub-rule in Figure 12 can check the similarities among multiple words that come after the selected numbers. Another variant can check the preceding word of numbers instead of the next word.

```
while for all $(q, a) \in S$ do
    if q contains no number or number related words then
        set this (q, a) pair to be ignored;
    end
    if q starts with the word "Where" then
        set this (q, a) pair to be ignored;
    end
    if q contains more than 40 words then
        set this (q, a) pair to be ignored;
    end
end
```

Figure 9. Example sub-rule 1.

```
create a dictionary D to hold the occurrence count of potential number pairs;
while for all (q, a) ∈ S such that its q is not set to be ignored do
    if q and a have at least 2 number values in total then
        find all potential number pairs for each number in the q and a group;
        increase the count of each number pair occurrence in the dictionary D;
    end
end
```

Figure 10. Example sub-rule 2.

```
assign (m1, m2) as the number pair with the max occurrence count;
if m1 appears before m2 in P then
    n2 slot ← the number in m1;
    n1 slot ← the number in m2;
else
    n1 slot ← the number in m1;
    n2 slot ← the number in m2;
end
```

Figure 11. Example sub-rule 3.

```
assign (m1, m2) as the number pair with the max occurrence count;
assign (m3, m4) as the number pair with the second highest occurrence count;
w1 ← the word following m1 in P ;
w2 ← the word following m2 in P ;
w3 ← the word following m3 in P ;
w4 ← the word following m4 in P ;
if w3 is a similar word compared to one of the words in the (w1, w2) pair then
    n3 slot ← the number in m3;
    n4 slot ← the number in m4;
end
if w4 is a similar word compared to one of the words in the (w1, w2) pair then
    n3 slot ← the number in m4;
    n4 slot ← the number in m3;
end
```

Figure 12. Example sub-rule 4.