# A History of Scheme for Word Problems

P. Reany

May 30, 2025

> You have to know what to look for, so you can spot it.
> — Papago Indian drug-enforcement border scout

## 1 Preface

I began *Scheme* in the 1980s as a means to systematically approach all college algebra type problems. In that time, I found many ways to increase clarity by various diagrammatic procedures, such as before-and-after diagrams, flow charts, tables, and others. There have been diagrammatic formalism in solving word problems for a long time before I entered the picture (such as in probability theory and age problems), so I certainly didn't invent them all. (Although I may have re-purposed some of them.) As for 'math in a flow chart', I got that idea directly from flow charting in computer programming. I figured that a good organizing principle in computer programming, should also be a good organizing principle in geometry and abstract algebra — and it is!

Also, I recently extended the formalism of *Scheme* to apply to problems in stoichiometry, which is a subject pervasive in chemistry. What brought me to adapt *Scheme* to stoichiometry was my desire (back in the day) to tutor chemistry in the near future. What brought me to develop *Scheme* originally was my frustration with the problem-solving methods I learned for algebra word problems in high school, so very long ago.

Finally, this article started as Appendix C in another, much longer, paper on stoichiometry, so you may see a capital 'C' here and there in the naming conventions, because I was too lazy to remove them.

## 2 Introduction

> Small moves, Ellie. Small moves.
> — from the movie, *Contact*

As I look back on it now, I think that my basic approach to solving word problem in high school, and for years beyond, was to:

1) Look for variables (unknowns).
2) Look for relationships (usually equations) on those variables.
3) Solve the system.

I'll refer to this set of heuristics as the *naive approach*.

Now, these heuristics are not actually wrong, but my experience over the last four decades has been that they aren't exactly efficient, either. It took me a long time to realize that my 'logically correct' heuristics just didn't seem to work — at least not for me. It remained for me a frustrating logical paradox.

But slowly over the years after I graduated college, I began to see a different way to approach these problems. Although I don't remember the details on how I came to resolve the impasse on algebra problem solving, I do remember that it all boiled down to answering just two logical questions:

1) How should one best define an algebra word problem? and
2) How should one build efficient heuristics upon that definition?

To the first question, I answer: An algebra word problem is the translation of an algebra problem (given in some natural language) into one or more equations (and/or inequalities), in one or more unknowns, and then solve for those unknowns. (For the purpose of introducing *Scheme*, I will ignore the complications of word problems with inequalities, because they don't add anything to the overall concepts I need to present.)

# 3 Heuristics

My single greatest insight was that virtually all equations in algebra word problems can be categorized into a handful of easily recognizable types. (And this is what I mean by to 'know what to look for' in the introductory quote.) The following are the most common types of **equation generators** I have found:
1) Every total is **equal** to the sum of its parts.
2) Every invariant $Q$ in a 'before-and-after' process generates the **equation** $Q_i = Q_f$, where $i$ is the initial state and $f$ is the final state.
3) A proportion: Every proportion claims the **equality** of two fractions (ratios).
4) Formulas (**equations**) from science or geometry, such $V = IR$ from physics, or area of a circle $= \pi r^2$ from geometry.
5) **Equations** that constrain the unknowns of the problem, but are not of the previous four types. I refer to such **equations** as *constituitive relations*, such as Sally's age = one more than twice John's age.

The next greatest insight I had in my search for an efficient plan to solve word problem was to **not** rush to find unknowns, but rather, after finding a word equation to translate, execute on it a *step-wise refinement* of the word equation until in its final form it exists as a pure algebraic statement. This procedure has

a name: it's called the *top-down-approach with step-wise refinement*[1] to problem solving (for those who remember the movie *Contact* — 'small moves').

# 4 Example Problems

For example, consider the word problem (often referred to as a 'rate' problem):

▶ Printer #1 can print a 100 copies of a document in 3.4 hours and Printer #2 can print out the same print job in 2.5 hours. How long will it take for the print job to complete if both printers work on the job together, starting and stopping at the same time?

Now, we resist the urge to rush in to find unknowns. Instead, first, we look for totals and parts. Are there any? (Look hard to find them!) Yes, there are — or, rather, there is. There is a total of one job being done by two contributing printers: Printer 1 and Printer 2. All right, we know that every total is **equal** to the sum of its parts. So, let's introduce the shorthand 'part of job done by' → PJDB. Then our highest-level equation is

$$1 \text{ job} = (\text{PJDB Printer 1}) + (\text{PJDB Printer 2})\,. \tag{1}$$

Generally speaking, the amount of production of a machine over time is given by the product of the rate $R$ at which it produces output $\times$ the time $T$ it runs. Therefore, let $R_1$ be the average rate at which Printer 1 can work, which is 1 job/3.4 hours.[2] Likewise, $R_2$ is the average rate at which Printer 2 can work, which is 1 job/2.5 hours.[3] Now, the most general expression we can write for the refinement of the last equation is (suppressing units)

$$1 = R_1 T_1 + R_2 T_2\,, \tag{2}$$

where $T_1$ and $T_2$ are the respective times that Printer 1 and Printer 2 are operating, which, in this particular problem, are the same number, which we'll just call $T$. So, the last equation becomes

$$1 = (R_1 + R_2)T\,. \tag{3}$$

So, now we can solve for time $T$, the time to finish the job:

$$T = \frac{1}{R_1 + R_2}\,. \tag{4}$$

Using the numbers given,

$$T = \frac{1}{(1/3.4) + (1/2.5)} = 1.44\,[\text{hours}]\,. \tag{5}$$

[1] This is a technique I learned for developing algorithms in a computer programming class.

[2] How to decide the units? Should it be job/hour or hour/job? It must be the former because we must have *job* in the numerator to match *job* being in the numerator on the left-hand side of the equation.

[3] We are employing the Zeroth Rule of Problem Solving to make the simplifying assumption that the average rate will be accurate for arbitrarily long or short time intervals.

Thus, solving the problem in 'small moves' makes it easier to solve. And, by the way, if one has to solve a similar problem in which the constituitive relation on $T_1$ and $T_2$ is more complicated than just $T_1 = T_2$, then that can be easily dealt with, too.

Consider the following problem, in which there is a non-trivial constituitive relation on the time variables of two people doing a cooperative job (often referred to as a 'work' problem):

▶ Steve can mow a lawn in three hours and Joe can mow the same lawn in two hours. How long will each of them take to mow the lawn if they both work on it together, except that Joe works 20 minutes before Steve starts to work?

Clearly, we won't be able to just mindlessly apply the so-called 'work formula' in (4) in this case, since $T_1 \neq T_2$. If we let $T_1$ be Joe's time and $T_2$ be Steve's time, then we can write down the constituitive relation

$$T_1 = T_2 + 1/3\,, \tag{6}$$

where time is measured in hours. However, this same relationship can be obtained by taking a 'total is the sum of its parts' analysis. (How? Hint: Use the top-down approach; partition the timeline.)

# 5 Example Problem with Diagram

I didn't employ a diagram for the previous problem, but I will for this one.

---

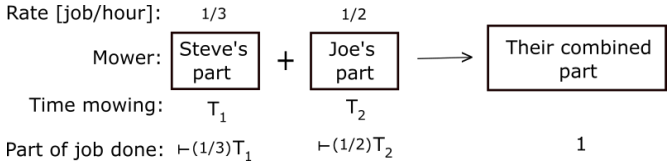


Figure C1. Diagramming this work problem as a total is the sum of its parts. The turnstile before a value or expression means that it was computed using only information in the same column.

---

Observation on the above figure: Viewed from the perspective of object-oriented programming, the boxes represent *objects* that have rates and times as data, or *properties*, and (from the bottom line) has the function (rate × time) as a *method*. The columns represent an *encapsulation* of an objects's data and the functions on that data. So, instead of placing relevant information about a given problem scattered about on the page, *Scheme*'s diagrammatic encapsulation keeps an object's related data together in one column.

All this fussiness about encapsulation is about presenting information to yourself and to someone else so that the information is as easy to comprehend

as is possible. English composition has similar rules to accomplish the same goal. As Struck and White prescribed one of those rules: "Keep related words together" (*The Elements of Style*).

# 6 The Development of *Scheme*: — My Thought Processes

Back in the mid-1980s, when I was first formulating the rules of *Scheme*,[4] I used a procedure to solve a problem similar to those just presented, and was able for the first time, not only to solve the problem, but also to feel as though I truly understood the solution. And *then* I knew that I had the beginnings of a scheme to solve algebra word problems with confidence. Since then, I have scoured textbooks and the Internet to find challenging/difficult word problems on which to test *Scheme*. And I admit, some of them *were* challenging to me. To date, I've written over 30 short unpublished papers using *Scheme* to solve algebra word problems, this series of stoichiometry papers not included.

I have a few general comments to make about *Scheme* at this point.

► Part of the success of *Scheme* comes from its combination of eclectic heuristics for solving word problems (already discussed) and its own terminology and diagramming protocols. Let's deal with the terminology first. If you're already familiar with current nomenclature on algebra word problems, no doubt you know of 'rate' problems, 'work' problems, and 'mixture' problems.

As an example of a mixture problem, consider the coin problem given on page [pagerefcoinproblem], in which we consider a pile of coins as a mixture of pile of nickels and a pile of dimes. The rates involved are the conversion factors that convert the unit coin of a given denomination to its dollar value.

Now, I'm not against having these terms ('rate', 'work', 'mixture'), but I did want *Scheme* to have a single term rule them all (and that has a nice ring to it). To that end, I invented (or, probably only reinvented) the term *Mixed-Rate* problem.

**Definition:** A *mixed-rate problem* is an algebra problem in which two or more 'machines' work together, at generally different rates, to accomplish a common goal.

The point is that this definition is designed to cover, all at once, rate problems, work problems, and mixture problems. But there's a cost to co-opting the word 'machine' for such abstraction, which is that often the things to which it applies will not look very much like a 'machine'.

**Definition:** A *simple machine* is a named entity in an algebra problem that converts one unit into another unit by a fixed rate, $R$, say, and $R$ is also said to

[4] The name *Scheme* is only a recent addition to the set of rules (heuristics), being adopted around 2015.

be a *conversion factor*. To every conversion factor $R$ is associated some quantity $Q$ with unit the same as the denominator unit of $R$.

Now a generalization:

**Definition:** A *machine* is a named entity in an algebra problem that has associated to it one or more conversion factors, $R_i$, each having its own associated quantity $Q_i$.

For example, Alice's nutritionist has recommended to her how many calories and grams of protein she should consume every morning. One morning she wants to fulfill these requirement with milk and a bagel. She has to calculate how much of each to consume. Her nutritionist has provided for her a table that contains for each of these food items the calories per ounce and the grams of protein per ounce. Thus, the 'machine' milk and the 'machine' bagel each have two rates and two quantities associated with them.

▶ One might ask at this point how one should define a formal representation of a *Scheme* machine. This is one way: The formal representation/denotation of a *Scheme* machine is an ordered $(2n+1)$-tuple, whose first entry is the name of the machine, the next $n$ entries are the names of the conversion factors (or rates), and the last $n$ entries are their respective associated quantities. For example, a problem in which two pumps, $P_1$ and $P_2$, cooperate to fill a tank could be represented by $(P_1, R_1, Q_1)$ and $(P_2, R_2, Q_2)$.

But where would I come up with such a bizarre abstract 'machine' concept? Well, it happened to me quite naturally as I solved hundreds of 'difficult' word problems over the years. They seemed to all morph into each other. For example: What's really the difference between two pumps (real machines) actively filling a tank at two different rates **with** two drain pipes passively emptying a tank at two different rates **with** two printers printing a print job together at two different rates **with** two painters painting a house together at two different rates **with** forming a coffee blend by adding together two different coffees costing two different amounts per pound **with** alloying two different metals together that cost different amounts per pound **with** combining two piles of coins, one nickels, one dimes, into a single collection of coins, each pile adding quantities in different amounts and contributing dollar values at different rates, and so on?

The point being, *not* that there is some metaphysical 'machineness' intrinsic to all these examples of pairs of cooperating things, but that in every case listed above that is *not* a pair of cooperating real machines, the structure of the layout of the solution is the same as in the cases where they *are* two cooperating real machines. In other words, there's an analogy or isomorphism between them. (Similar to the point of category theory, is it?)

So, I state a rule that is already well known in mathematics, though I'm not aware that it has a name: There are times in the analysis of a problem that it's convenient to treat different things as the 'same' and to treat similar things as different. I can readily think of examples from combinatorics. I have an even simpler example: The claim that the operation of 3 apples + 2 oranges

is meaningless is to emphasize the difference between apples and oranges, yet, to claim that the operation 3 apples + 2 oranges is meaningful and is equal to 5 pieces of fruit is to emphasize their commonality.

▶ Now, on to *Scheme* diagramming protocols: There are plenty of examples of how I diagram word problems in *Scheme* in the worked example problems presented in the beginning of this paper. The protocol is basically this: Entities are represented by boxes, whose names are in the boxes. Rates of change (conversion factors) are listed above the boxes and quantities are listed below the boxes. Rates of change include fractional amounts and ratios.[5] When a fractional amount is in the form of a part-to-whole, it has no units.

I found a similar diagramming protocol to that of *Scheme* at the website

`http://mgccc.edu/learning_lab/math/alg/howtomix.pdf`

# 7 Tabular Forms Useful in *Scheme*

One last point on *Scheme* diagrams: Although it seems to be the popular thing to do these days in both algebra word problems and stoichiometry to place variables and expressions in tabular form, I generally prefer not to, though there are some exceptions. Obviously, I prefer the boxes-and-arrows approach. But there is one notable exception: That being the so-called 'age problem' and their generalization (the 'temporal' problem).

In a typical age problem, two constituitive relations are given on the ages of two people, one on their current ages and another on their ages a given number of years in the past or in the future. The end result is immediately a system of two equations in two unknowns, which is ready to be solved.[6]

The following are problems that *Scheme* recommends using the tabular form of diagramming. This first one is found at

`https://www.algebra.com/algebra/homework/word/age/Age-problems-and-their-solutions.lesson`

▶ **Problem 1:** Kevin is 4 years older than Margaret.
Next year Kevin will be 2 times as old as Margaret.
How old is Kevin?

[5] The extension of *Scheme* to deal with stoichiometry adds a twist to this procedure: The *MoleStats line* is placed immediately below the boxes.

[6] I should also admit that I find the tabular form of the representation of variables and expressions in probability to be of great value to me.

**Solution:** Use a tabular form of a diagram:

| Person / Age | Kevin | Margaret | Constituitive Relations |
|---|---|---|---|
| Now | $K$ | $M$ | $K = M + 4$ |
| Year from now | $K + 1$ | $M + 1$ | $(K + 1) = 2(M + 1)$ |

Figure C2. Using a tabular form for the diagram. The double lines are there to separate off the constituitive relations from the Person/Age information. Compared to a nontabular form of solution to this problem, the tabular form is definitely CNO (Clean, Neat, Organized), which aids in problem solving.

Regarding the diagram above, the two constituitive equations shown there *is* the system of equations to be solved for $K$ and $M$, yielding, $K = 7$ and $M = 3$.

Now, I'd like to say a word about how some people approach problems of this type. They will adopt what I call the *accelerated substitution* mode, whereby, they'll introduce the variable, $K$ for Kevin's age, and then the variable $K-4$ for Margaret's age. This isn't technically wrong, of course, but I think it's confusing for beginners to learn. We shouldn't confuse for the beginner the distinction between the basic variables to a problem and the constituitive relations on those variables. Furthermore, accelerated substitution doesn't generalize well. For example, how does it work when the problem has three or more basic variables to solve for? However, I'm not totally against accelerated substitution. I just think that the student should learn the general concepts first, and the shortcuts later.

The next example problem I'd like to show is found at

`https://www.basic-mathematics.com/hard-word-problems-in-algebra.html`

**Problem 99.** Jacob's hourly wage is 4 times as much as Noah. When Jacob got a raise of 2 dollars, Noah accepted a new position that pays him 2 dollars less per hour. Jacob now earns 5 times as much money as Noah. How much money do they make per hour after Jacob got the raise?

**Solution**

This kind of problem presents itself as a generalization of of the age problem. I'll refer to the class of problems that include this kind of problem and the age problems as *temporal problems*, where either time itself (ages, of course, have the units of time) or the main variables are functions of time.[7]

[7]By 'functions of time' I do not mean to imply a continuous function of time, though it might be. For example, it could be that on a time line, all one knows is that something is true at one time and something else is true 'at a later time'.

Let's begin with a tabular diagram:

| | Jacob | Noah | Constituitive Relations |
|---|---|---|---|
| Beginning Wages | $J$ | $N$ | $J = 4N$ |
| Ending Wages | $J + 2$ | $N - 2$ | $J + 2 = 5(N - 2)$ |

Figure C3. As a general rule, I avoid placing algebraic information into tables, but in this kind of problem (a 'temporal' problem), a table works very well.

As in the previous problem, we find the solution by simultaneously solving the constituitive relations presented in the diagram, obtaining (in units of dollars per hour) $J = 48$ and $N = 12$. Therefore, after the wages change:

$$J + 2 = 50\,, \quad N - 2 = 10\,. \tag{7}$$

As a final comment on temporal problems, I'd like to point out that in common parlance, we could categorize them as 'before-and-after' problems (which could cause some confusion). To obviate this, in *Scheme* this designation is used only for such problems as can generate a nontrivial equation of the form $Q_f = Q_i$ (or $Q_i = Q_f$), where $Q$ is some invariant quantity of the process.

# 8 Conclusion

I have on this site many worked problems in college algebra, GMAT problems, stoichiometry problems, and others — all using *Scheme*. And I can attest that there are some really difficult word problems out there that are "just algebraic" in nature at the college level. But that don't mean they ain't hard.

# References


[1] P. Atkins and L. Jones. *Chemical Principles: The Quest for Insight*, 3rd Ed. Freeman (2005).

[2] R. Blitzer. *Intermediate Algebra for College Students*, 3rd Ed. Prentice-Hall (2002).

[3] M. Hein and S. Arena *Foundations of College Chemistry*, alternate 12th ed, John Wiley & Sons (2007), 421–422.

[4] H. Rolf. *Finite Mathematics*, 5th Ed. Brooks/Cole (2002), p. 57.

[5] M. S. Silberberg. *Chemistry: The Molecular Nature of Matter and Change* 4th Ed. McGraw-Hill (2006).