Chapter 5: The Finite Domain Solver

H. Simonis
4C, University College Cork, Ireland
h.simonis@4c.ucc.ie

July 31, 2008

Abstract

In this part, we introduce the finite domain solver of the ECLiPSe system by way of examples. We start with a crypt-arithmetic puzzle, the well known SEND+MORE=MONEY problem, to introduce the basic concepts of domain variables and constraints, propagation and search.

In a second part, we look at the Sudoku puzzle, a Japanese number puzzle which introduces the ideas of global constraints, consistency levels and heuristic search methods.

Contents

1 Problem

We begin with the definition of the SEND+MORE=MONEY puzzle. It is often shown in the form of a hand-written addition:

$$\begin{array}{ccccc} & \mathrm{S} & \mathrm{E} & \mathrm{N} & \mathrm{D} \\ + & \mathrm{M} & \mathrm{O} & \mathrm{R} & \mathrm{E} \\ \hline \mathrm{M} & \mathrm{O} & \mathrm{N} & \mathrm{E} & \mathrm{Y} \end{array}$$

This is an example of a crypt-arithmetic puzzle, and is one of the classical, early examples of constraint programming. The puzzle is defined by the following rules:

- Each character stands for a digit from 0 to 9.
- Numbers are built from digits in the usual, positional notation.
- Repeated occurrence of the same character denote the same digit.
- Different characters denote different digits.
- Numbers do not start with a zero.
- The equation must hold.

2 Model

If we want to model this problem as a finite domain constraint problem, we have to define variables and constraints. *Variables* range over *domains*, which describe the possible values they can take. In this section we consider only finite domains, indeed only finite domains expressed as finite subsets of natural numbers. We will consider constraint problems with infinite domains in a later section. An *assignment* is a mapping of the variables to values in their respective domains.

A *constraint* ranges over one or multiple variables and expresses a condition which must hold between these variables. Constraints can take many forms, they may be explicit, describing combinations of values that are allowed or excluded, or symbolic, for example in the form of arithmetic constraints. Formally, we can see a constraint as a subset of the Cartesian product of the domains of its variables. An assignment *satisfies* a constraint if its projection on the variables of the constraint belong to this constraint subset.

An assignment of values to variables is a *solution* if all constraints are satisfied.

For many problems, we have considerable choices in selecting the variables and constraints of our problem. Often, models differ dramatically in the number of variables needed, the complexity and number of constraints used to describe the problem, and the speed by which a constraint solver can find a solution.

For the small puzzle considered here, the choice of the variables is fairly obvious: Each character is a variable, which ranges over the values 0 to 9.

For the constraints, a natural description would be:

Listing 1: Program Sendmory

```
1  :-module(sendmory).
2  :-export(sendmory/1).
3  :-lib(ic).
4
5  sendmory(L):-
6          L = [S,E,N,D,M,O,R,Y],
7          L :: 0..9,
8          alldifferent(L),
9          S #\= 0,
10         M #\= 0,
11         1000*S + 100*E + 10*N + D +
12         1000*M + 100*O + 10*R + E #=
13         10000*M + 1000*O + 100*N + 10*E + Y,
14         labeling(L).
```

- An *alldifferent* constraint between all variables, which state that two different variables must have different values. This is a very common constraint, which we will encounter in many other problems later on.
- Two *disequality constraints* (variable X must be different from value V) stating that the variables at the beginning of a number can not take the value 0.
- An arithmetic *equality constraint* linking all variables with the proper coefficients and stating that the equation must hold.

Remember that this is not the only model of this problem, we will discuss some alternatives later on.

3 Program

The program shown in Listing 1 is an example of a finite domain constraint program in ECLiPSe, which implements our model above.

Line 1 shows the module declaration. Each file should be a module, and there should be one module per file. All predicates defined in a module are only visible inside the module, unless they are exported.

Line 2 shows an export directive. It states that the predicate sendmory with arity one (with one argument) should be exported from this module.

Line 3 contains a library directive. We are loading the *ic* library, which defines the *I*nterval *C*onstraints in the ECLiPSe language.

Lines 5-14 define the sendmory predicate which is the core of our model. It consists of a single clause with the clause head `sendmory(L)`. The clause has a single argument L, which is a list of variables as defined in line 6. The square brackets denote lists in Prolog.

We next define the domain of the variables in line 7. The infix operator `::` takes a variable or list of variables as the first (left) argument, and a domain (here 0..9) as the second (right) argument.

We then express the *alldifferent* constraint in line 8. This constraint is a built-in constraint in the ic library, and is thus shown in bold.

Lines 9-10 express the two disequality constraints. The ic library defines the operator `#\=` for this purpose.

The big linear equality constraint is expressed in line 12. The operator `#=` is used in the ic library.

Finally, we call the builtin search routine *labeling* to assign values to the variables.

4 Running the program

To run the program, we have to enter the query

```
sendmory:sendmory(L).
```

which says we want to run query `sendmory(L)` in the module *sendmory*. The program returns the answer

```
L = [9, 5, 6, 7, 1, 0, 8, 2]
yes (0.00s cpu, solution 1, maybe more)
```

This is a solution to the puzzle, as a quick check will show.

But how did the program come up with this solution?

5 Constraint Setup

We will now discuss how the ECLiPSe program did find the solution. For this, we will run through the program step by step, and explain the reasoning of the constraint engine at each step. The presentation is not intented to explain the details of the implementation, but should rather show the reasoning involved, presented in a human readable form.

5.1 Domain Definition (line 7):

$$[S,E,N,D,M,O,R,Y] \in \{0..9\}$$

Figure 1 shows the state of the domain variables at this point. Each line shows one variable, each column one value. Initially, all values are allowed for all variables, this is indicated by white cells.

5.2 Setup of alldifferent (line 8)

When the alldifferent constraint is first called, all variables have the same domain, none of the variables are assigned. At this point, the constraint can not do any propagation, it therefore just suspends and waits until one of its variables is changed either by other constraints or as part of the search routine.

	0	1	2	3	4	5	6	7	8	9
S										
E										
N										
D										
M										
O										
R										
Y										

Table 1: Initial Domains

5.3 Disequality constraints (lines 9-10):

The program now handles the two disequality constraints. As each constraint only deals with a single variable, we can solve them by removing the value from the domain of the variable. This leads to an update in two of the domains:

$$S \in \{1..9\}, M \in \{1..9\}$$

	0	1	2	3	4	5	6	7	8	9
S	▨									
E										
N										
D										
M	▨									
O										
R										
Y										

The two disequality constraint are now solved, they can be discarded.

Table 2 shows the state of the variables after these steps, only two cell have been colored gray to indicate that the values have been removed.

	0	1	2	3	4	5	6	7	8	9
S	▨									
E										
N										
D										
M	▨									
O										
R										
Y										

Table 2: After Disequality Constraints

5.4 Equality Constraint

The next constraint encountered is the equality constraint in line 12. The processing of this constraint is more complex, we start with a normalisation of the constraint.

5.4.1 Normalization

The constraint as stated contains multiple occurrences of the same variable. In a first step, we reduce these multiple occurences by adding their coefficients. For didactic purposes, we avoid negative coefficients by placing terms both on the left and the right hand side of the equation.

Original equation: The original constraint, written in tabular form, looks like this.

	1000*S+	100*E+	10*N+	D
	+1000*M+	100*O+	10*R+	E
10000*M+	1000*O+	100*N+	10*E+	Y

Variable M occurs twice: In a first step, we simplify occurences of variable M.

	1000*S+	100*E+	10*N+	D
	+**1000*M**+	100*O+	10*R+	E
10000*M+	1000*O+	100*N+	10*E+	Y

Simplified equation:

	1000*S+	100*E+	10*N+	D
	+	100*O+	10*R+	E
9000*M+	1000*O+	100*N+	10*E+	Y

Variable O occurs twice: Next we treat variable O. Note that the normalization is not implemented this way in the constraint solver, we just want to show each step individually.

	1000*S+	100*E+	10*N+	D
	+	**100*O**+	10*R+	E
9000*M+	**1000*O**+	100*N+	10*E+	Y

Simplified equation:

	1000*S+	100*E+	10*N+	D
		+	10*R+	E
9000*M+	**900*O**+	100*N+	10*E+	Y

Variable N occurs twice: Variable N also occurs twice.

	1000*S+	100*E+	**10*N**+	D
		+	10*R+	E
9000*M+	900*O+	**100*N**+	10*E+	Y

Simplified Equation:

	1000*S+	100*E+		D
		+	10*R+	E
9000*M+	900*O+	**90*N**+	10*E+	Y

Variable E occurs three times: The last variable to be handled is variable E.

	1000*S+	**100*E**+		D
		+	10*R+	**E**
9000*M+	900*O+	90*N+	**10*E**+	Y

Simplified equation:

	1000*S+	91*E+		D
		+	10*R	
9000*M+	900*O+	90*N+		Y

Final equation: The final version of the equation can be written in a single line:

$$1000 * S + 91 * E + 10 * R + D = 9000 * M + 900 * O + 90 * N + Y$$

5.4.2 Propagation

We now look at the initial propagation of the equality constraint, while noting that the alldifferent constraint is suspended, but will be woken when variables are fixed to values.

Propagation of equality The reasoning of the equality constraint is based on an evaluation of the left and right hand of the equation. We can do this by checking the domains of all variables involved. In the following, the notation $X^{l..h}$ denotes the variable X with a current domain of $l..h$.

$$\begin{aligned} &1000 * S^{1..9} + 91 * E^{0..9} + 10 * R^{0..9} + D^{0..9} = \\ &\quad 9000 * M^{1..9} + 900 * O^{0..9} + 90 * N^{0..9} + Y^{0..9} \end{aligned}$$

We compute lower and upper bounds of each side of the equation based on the domains and the coefficients of the variables. This leads to the following intervals:

$$\begin{aligned} &\underbrace{1000 * S^{1..9} + 91 * E^{0..9} + 10 * R^{0..9} + D^{0..9}}_{1000..9918} = \\ &\quad \underbrace{9000 * M^{1..9} + 900 * O^{0..9} + 90 * N^{0..9} + Y^{0..9}}_{9000..89919} \end{aligned}$$

If the equation holds, then both sides must be equal. We can therefore use the intersection of the intervals as the range of values that we have to consider for the equation:

$$\begin{aligned} &\underbrace{1000 * S^{1..9} + 91 * E^{0..9} + 10 * R^{0..9} + D^{0..9}}_{9000..9918} = \\ &\quad \underbrace{9000 * M^{1..9} + 900 * O^{0..9} + 90 * N^{0..9} + Y^{0..9}}_{9000..9918} \end{aligned}$$

We now check for each variable if their lower resp. upper bounds are consistent with that value range. For example, M can be atmost 1, otherwise its contribution would exceed the range limit set by the equation. In a similar way, S must be at least 9, otherwise the left hand side can not reach the lower bound of 9000. Finally, the variable O can be atmost 1.

This leads to the following domain reductions:

$$M = 1, S = 9, O \in \{0..1\}$$

Propagation of alldifferent: This assignment wakes up the alldifferent constraint, which further reduces the domains of the variables.

$$O = 0, [E, R, D, N, Y] \in \{2..8\}$$

Resulting equation: The domain changes wake up the equality constraint again. It now takes the following form:

$$1000 * 9 + 91 * E^{2..8} + 10 * R^{2..8} + D^{2..8} = 9000 * 1 + 900 * 0 + 90 * N^{2..8} + Y^{2..8}$$

Removal of constants We first remove the constant values in the equality constraint, this makes the example easier to follow.

$$91 * E^{2..8} + 10 * R^{2..8} + D^{2..8} = 90 * N^{2..8} + Y^{2..8}$$

Propagation of equality:

$$\underbrace{91 * E^{2..8} + 10 * R^{2..8} + D^{2..8}}_{204..816} = \underbrace{90 * N^{2..8} + Y^{2..8}}_{182..728}$$

$$\underbrace{91 * E^{2..8} + 10 * R^{2..8} + D^{2..8} = 90 * N^{2..8} + Y^{2..8}}_{204..728}$$

$$N > 2, E < 8$$

Propagation of equality: Each change of the domains of the variables triggers a re-execution of the equality constraint.

$$91 * E^{2..7} + 10 * R^{2..8} + D^{2..8} = 90 * N^{3..8} + Y^{2..8}$$

$$\underbrace{91 * E^{2..7} + 10 * R^{2..8} + D^{2..8}}_{204..725} = \underbrace{90 * N^{3..8} + Y^{2..8}}_{272..728}$$

$$\underbrace{91 * E^{2..7} + 10 * R^{2..8} + D^{2..8} = 90 * N^{3..8} + Y^{2..8}}_{272..725}$$

$$E > 2$$

Propagation of equality:

$$91 * E^{3..7} + 10 * R^{2..8} + D^{2..8} = 90 * N^{3..8} + Y^{2..8}$$

$$\underbrace{91 * E^{3..7} + 10 * R^{2..8} + D^{2..8}}_{295..725} = \underbrace{90 * N^{3..8} + Y^{2..8}}_{272..728}$$

$$\underbrace{91 * E^{3..7} + 10 * R^{2..8} + D^{2..8} = 90 * N^{3..8} + Y^{2..8}}_{295..725}$$

$$N > 3$$

Propagation of equality:

$$91 * E^{3..7} + 10 * R^{2..8} + D^{2..8} = 90 * N^{4..8} + Y^{2..8}$$

$$\underbrace{91 * E^{3..7} + 10 * R^{2..8} + D^{2..8}}_{295..725} = \underbrace{90 * N^{4..8} + Y^{2..8}}_{362..728}$$

$$\underbrace{91 * E^{3..7} + 10 * R^{2..8} + D^{2..8} = 90 * N^{4..8} + Y^{2..8}}_{362..725}$$

$$E > 3$$

Propagation of equality:

$$91 * E^{4..7} + 10 * R^{2..8} + D^{2..8} = 90 * N^{4..8} + Y^{2..8}$$

$$\underbrace{91 * E^{4..7} + 10 * R^{2..8} + D^{2..8}}_{386..725} = \underbrace{90 * N^{4..8} + Y^{2..8}}_{362..728}$$

$$\underbrace{91 * E^{4..7} + 10 * R^{2..8} + D^{2..8} = 90 * N^{4..8} + Y^{2..8}}_{386..725}$$

$$N > 4$$

Propagation of equality:

$$91 * E^{4..7} + 10 * R^{2..8} + D^{2..8} = 90 * N^{5..8} + Y^{2..8}$$

$$\underbrace{91 * E^{4..7} + 10 * R^{2..8} + D^{2..8}}_{386..725} = \underbrace{90 * N^{5..8} + Y^{2..8}}_{452..728}$$

$$\underbrace{91 * E^{4..7} + 10 * R^{2..8} + D^{2..8} = 90 * N^{5..8} + Y^{2..8}}_{452..725}$$

No further domain reduction is possible at this time. The propagation stops and the program continues with its normal execution. Figure 3 shows the domains of the variables after the initial setup.

	0	1	2	3	4	5	6	7	8	9
S	▨	▨	▨	▨	▨	▨	▨	▨	▨	🟥
E	▨	▨	▨	▨					▨	▨
N	▨	▨	▨	▨	▨					▨
D	▨	▨								▨
M	▨	🟥	▨	▨	▨	▨	▨	▨	▨	▨
O	🟥	▨	▨	▨	▨	▨	▨	▨	▨	▨
R	▨	▨								▨
Y	▨	▨								▨

Table 3: Domains after setup

6 Search

The predicate

`labeling([S,E,N,D,M,O,R,Y])`

in line 14 on page 4 will enumerate the variables in the given order, each time starting with the smallest value in the domain. If the assignment of that value fails, the next value will be tried, and so on. If no more values remain, the search will backtrack to the last previous choice and try its next alternative. This will continue until either

- a valid assignment is found for all variables, and the search *succeeds*
- no more untested choices remain, the search *fails*

6.1 Step 1

The first variable S is aleady assigned to value 9. Figure 1 shows the search

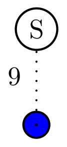

Figure 1: Search Tree Step 1

tree at this point. The top node shows the variable being assigned (here S), the label on the edge shows the value that is taken (here 9). The link is dotted, as the variable is already assigned, and no new choice is required.

6.2 Step 2

6.2.1 Alternative $E = 4$

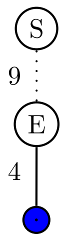

Figure 2: Search Tree Step 2, Alternative 1

The next variable is E, its smallest value is 4. Figure 2 shows the search tree at this point. We are currently assigning variable E to value 4. As this is a proper choice, a solid link is used.

Assigning E to 4 will wake the equality constraint

Propagation of equality:

$$91 * 4 + 10 * R^{2..8} + D^{2..8} = 90 * N^{5..8} + Y^{2..8}$$

$$\underbrace{91 * 4 + 10 * R^{2..8} + D^{2..8}}_{386..452} = \underbrace{90 * N^{5..8} + Y^{2..8}}_{452..728}$$

$$\underbrace{91 * 4 + 10 * R^{2..8} + D^{2..8} = 90 * N^{5..8} + Y^{2..8}}_{452}$$

$$N = 5, Y = 2, R = 8, D = 8$$

Propagation of alldifferent: This constraint fails (can't have $R = 8$, $D = 8$)

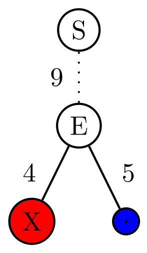

Figure 3: Search Tree Step 2, Alternative 2

6.2.2 Alternative $E = 5$

We backtrack to the last choice ($E = 4$), and then try the next value ($E = 5$). This situation is depicted in figure 3. The left child of node E is marked as a failure, we are currently testing value 5, the second alternative. The assignment of value ($E = 5$) will trigger both the alldifferent and the equality constraint. Assume that the alldifferent is woken first[1].

Propagation of alldifferent:

$$N > 5$$

Propagation of equality:

$$91 * 5 + 10 * R^{2..8} + D^{2..8} = 90 * N^{6..8} + Y^{2..8}$$

$$\underbrace{91 * 5 + 10 * R^{2..8} + D^{2..8}}_{477..543} = \underbrace{90 * N^{6..8} + Y^{2..8}}_{542..728}$$

$$\underbrace{91 * 5 + 10 * R^{2..8} + D^{2..8} = 90 * N^{6..8} + Y^{2..8}}_{542..543}$$

$$N = 6, Y \in \{2, 3\}, R = 8, D \in \{7..8\}$$

Propagation of alldifferent:

$$D = 7$$

Propagation of equality:

$$91 * 5 + 10 * 8 + 7 = 90 * 6 + Y^{2..3}$$

$$\underbrace{91 * 5 + 10 * 8 + 7}_{542} = \underbrace{90 * 6 + Y^{2..3}}_{542..543}$$

$$\underbrace{91 * 5 + 10 * 8 + 7 = 90 * 6 + Y^{2..3}}_{542}$$

$$Y = 2$$

[1]Which of the constraints is woken up first can be difficult to predict without understanding the details of the implementation of the solver

6.3 Further Steps

All variables have been assigned, the search succeeds. Figure 4 shows the resulting search tree. There is only a single variable (E) for which an actual choice had to be taken. The last node at the bottom of the tree is a green success node, indicating that a solution was found.

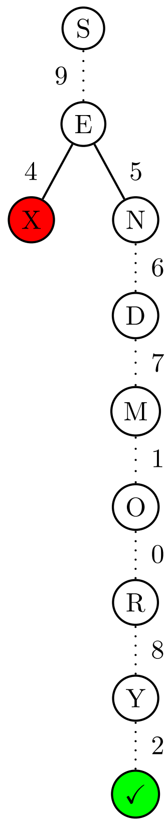

Figure 4: Search Tree, First Solution

Table 4 shows the state of the domain visualizer after the choice $E = 5$. All variables are assigned to a value, we can also see that the *alldifferent* constraint is satisfied, as every column contains atmost one assigned variable.

7 Solution

Taking all the information collected together, we find that the solution of the puzzle is:

$$\begin{array}{cccccc} & & 9 & 5 & 6 & 7 \\ + & & 1 & 0 & 8 & 5 \\ \hline & 1 & 0 & 6 & 5 & 2 \end{array}$$

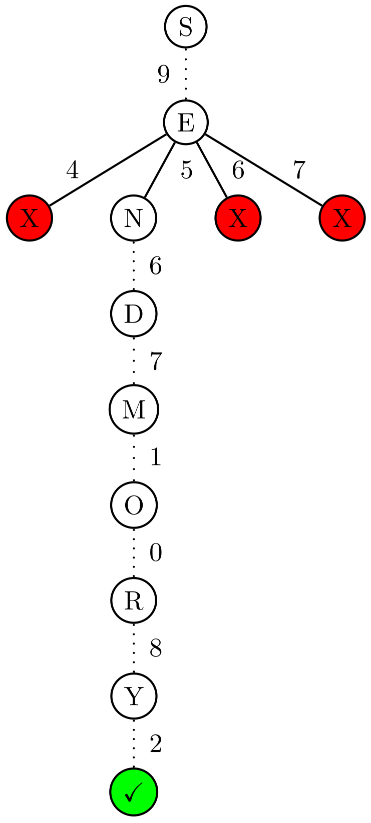

Figure 5: Complete Search Tree

	0	1	2	3	4	5	6	7	8	9
S	▨	▨	▨	▨	▨	▨	▨	▨	▨	🟥
E	▨	▨	▨	▨	▨	🟥	▨	▨	▨	▨
N	▨	▨	▨	▨	▨	▨	🟥	▨	▨	▨
D	▨	▨	▨	▨	▨	▨	▨	🟥	▨	▨
M	▨	🟥	▨	▨	▨	▨	▨	▨	▨	▨
O	🟥	▨	▨	▨	▨	▨	▨	▨	▨	▨
R	▨	▨	▨	▨	▨	▨	▨	▨	🟥	▨
Y	▨	▨	🟥	▨	▨	▨	▨	▨	▨	▨

Table 4: Domains after search

Listing 2: Program Sendmory without Disequality Constraints

```
:-module(sendmory).
:-export(sendmory/1).
:-lib(ic).

sendmory(L):-
        L = [S,E,N,D,M,O,R,Y],
        L :: 0..9,
        alldifferent(L),
        1000*S + 100*E + 10*N + D +
        1000*M + 100*O + 10*R + E #=
        10000*M + 1000*O + 100*N + 10*E + Y,
        labeling(L).
```

8 Lessons Learned

- Constraint models are expressed by variables and constraints.
- Problems can have many different models, which can behave quite differently. Choosing the best model is an art.
- Constraints can take many different forms.
- Propagation deals with the interaction of variables and constraints.
- Constraints only communicate via shared variables.
- Propagation usually is not sufficient, search may be required to find a solution.
- Propagation is data driven, and can be quite complex even for small examples.
- The default search uses chronological depth-first backtracking, systematically exploring the complete search space.
- The search choices and propagation are interleaved, after every choice some more propagation may further reduce the problem.

9 Model Variants (*)

9.1 Removing Disequality Constraints

9.2 Carry Variables with Multiple Equations

9.3 0/1 Variable Model

10 Improved Propagation (*)

	0	1	2	3	4	5	6	7	8	9
S										
E										
N										
D										
M			▨	▨	▨	▨	▨	▨	▨	▨
O										
R										
Y										

Table 5: Without Disequality Constraints: Domains After Setup

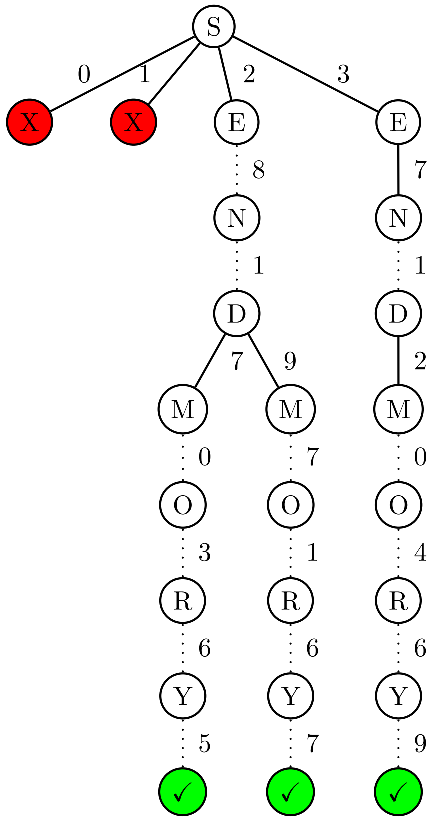

Figure 6: Without Disequality Constraints: First Three Solutions

Listing 3: Using Carry Variables and Multiple Equations

```
:- module(sendmory2).
:-export(sendmory/1).
:-lib(ic).

sendmory(L):-
          L=[S,E,N,D,M,O,R,Y],
          L :: 0..9,
          [C2,C3,C4,C5] :: 0..1,
          alldifferent(L),
          S #\= 0,
          M #\= 0,
          M #= C5,
          S+M+C4 #= 10*C5+O,
          E+O+C3 #= 10*C4+N,
          N+R+C2 #= 10*C3+E,
          D+E    #= 10*C2+Y,
          labeling(L).
```

	0	1	2	3	4	5	6	7	8	9
S	▨	▨	▨	▨	▨	▨	▨	▨	▨	🟥
E	▨	▨								▨
N	▨	▨								▨
D	▨	▨								▨
M	▨	🟥	▨	▨	▨	▨	▨	▨	▨	▨
O	🟥	▨	▨	▨	▨	▨	▨	▨	▨	▨
R	▨	▨								▨
Y	▨	▨								▨

Table 6: Sendmory2: Domains after setup

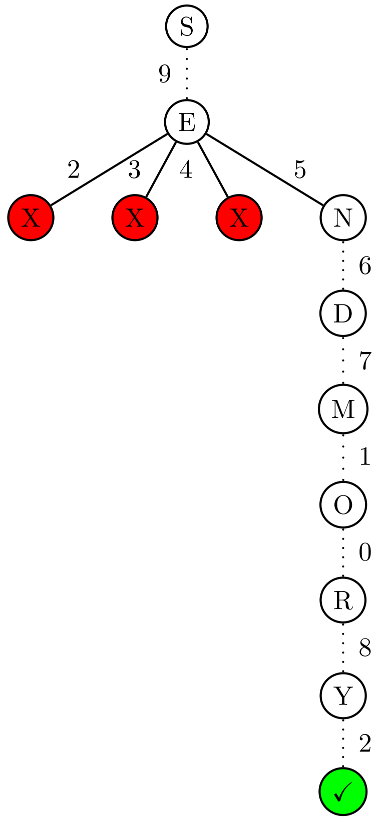

Figure 7: Sendmory2: First Solution

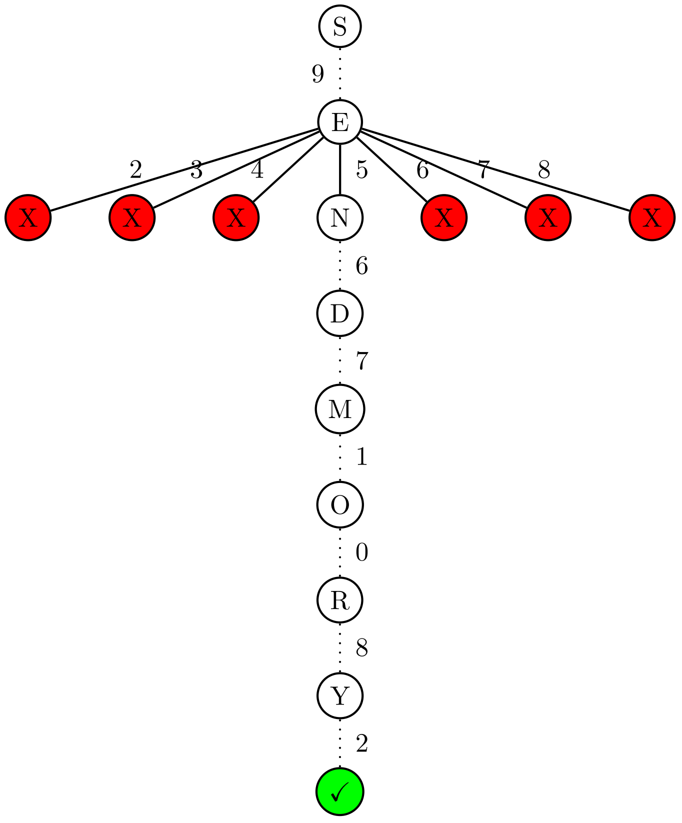

Figure 8: Sendmory2: Complete Search Tree