The Pruning of the Search Tree

A Presentation by Prof. Dr. E.W. Dijkstra

Sander Kooijmans

Eindhoven, August 2000

Contents

Summary

Problem solving can be represented as a search tree. In this report we show three techniques to prune this search tree, and thereby reducing the number of case analyses to be made: abstraction from irrelevant details, strengthening the problem, and application of simple theories. These techniques are illustrated by solving three simple problems, hence we can focus on the techniques and not on the problems. We also apply these techniques to solve a problem taken from a formal methods course. The examples show that application of these techniques indeed reduces the number of case analyses. If we omit choices where we can see easily that they do not lead to a solution, then application of these techniques delivers us a unique and generic solution.

1. Introduction

Problem solving can be represented as a **search tree**. Assume that problem solving is a process of searching for a solution. This process generally involves taking decisions. A search tree representing this process has the decisions as nodes. Each node has the alternatives as its children. Each alternative leads to another decision, is a dead end, or is a solution to the problem. **Case analysis** is considering the alternatives.

This report describes the contents of the presentation given by Prof. Dr. E.W. Dijkstra in honour of the eighteenth *Dies Natalis* of Gewis, on July 7, 2000. Gewis is the Eindhoven University community of mathematics and computer science students.

In his presentation Prof. Dr. E.W. Dijkstra showed three techniques to reduce the number of case analyses, by pruning the search tree: abstraction from irrelevant details, strengthening the problem, and application of simple theories. He illustrated these techniques by solving simple problems everybody knows from primary school, enabling everybody to focus on the techniques and not on the problems.

Chapters 2 to 4 describe the three techniques. In chapter 5 we apply the techniques to a problem taken from a formal methods course. This problem is not an example from Dijkstra's presentation.

This report is intended for people who have a basic knowledge of mathematics.

2. Abstraction from irrelevant details

The first technique we introduce is abstraction from irrelevant details. We illustrate this technique by solving the **cabbage-goat-wolf** problem.

2.1. The cabbage-goat-wolf problem

Consider a shepherd who has three **items**: a cabbage, a goat, and a wolf. With these items he wants to cross a river from the left bank to the right bank. He has a boat at his disposal to cross the river, but this boat can only carry the shepherd and at most one of his items.

The problem is that the shepherd can not leave the goat and the cabbage alone (on the same riverbank), because then the goat will eat the cabbage. Also he can not leave the wolf and the goat alone, because then the wolf will eat the goat.

How does the shepherd get his three items to the other side of the river, without the animals eating one another item?

2.2. The solution

We start solving this problem by drawing a dependency graph of the items. The items are the nodes of the graph, and an arrow from item *a* to item *b* denotes that item *a* will eat item *b* if the shepherd leaves them alone. The graph is depicted in Figure 1.

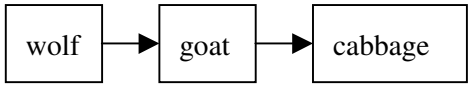

Figure 1 The directed dependency graph for the three items

Note that for the cabbage-goat-wolf problem the direction of the arrows is irrelevant. The shepherd can not transport the three items if he loses one of them. Therefore, it does not matter whether the wolf eats the goat or the goat eats the wolf, and whether the goat eats the cabbage or the cabbage eats[1] the goat.

Because the direction of the arrows in Figure 1 is irrelevant, we can replace them by undirected edges. The resulting graph is shown in Figure 2.

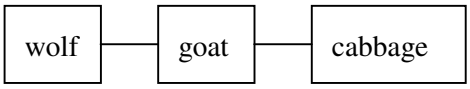

Figure 2 The undirected dependency graph for the three items

Figure 2 is symmetric about the goat. This symmetry indicates that the distinction between the wolf and the cabbage is irrelevant. It does not matter whether we leave the goat with the wolf or we leave the goat with the cabbage; in both cases the shepherd will

[1] We leave this to the reader's imagination.

lose one of his items. We abstract from the distinction between wolf and cabbage by replacing these items by α's. The result is shown in Figure 3.

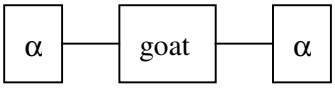

Figure 3 The symmetric dependency graph for the three items

From Figure 3 we conclude that the shepherd can only leave the two α's alone on the same riverbank.

We can write down a generic solution to the problem if we take some trivial rules into account: The shepherd never leaves the goat alone with an α, and the shepherd never performs redundant steps (*i.e.*, he never returns to a state that has already occurred.) A **state** is a pair of sets: The first set element is the set of items on the left riverbank, the second element is the set of items on the right riverbank. Also the shepherd is an element of (exactly) one of these sets.

The solution to the problem is shown in Derivation 1. A **derivation** is a list of states. Two successive states are separated by a transition. A **transition** describes in which direction the shepherd takes the boat, and which item he carries. A transition is denoted by an arrow pointing in the direction the shepherd sails. If the shepherd carries an item in the boat, the name of this item is written as a subscript of the arrow. Further we write comments between braces behind the arrow to justify the transitions.

({shepherd,goat,α,α} , ∅)
$\rightarrow_{goat}$ { The shepherd must leave the α's together, or else he leaves the goat with an α. }
({α,α} , {shepherd,goat})
$\leftarrow$ { If the shepherd takes back the goat, the results is the initial state. }
({shepherd,α,α} , {goat})
$\rightarrow_{\alpha}$ { If the shepherd does not take an α, then the result is a state that has already occurred. }
({α} , {shepherd,goat, α})
$\leftarrow_{goat}$ { If the shepherd does not take the goat, he loses an item. }
({shepherd,goat,α} , {α})
$\rightarrow_{\alpha}$ { If the shepherd takes the goat, then the result is a state that has already occurred. }
({goat} , {shepherd,α,α})
$\leftarrow$ { If the shepherd takes back an α, then the result is a state that has already occurred. }
({shepherd,goat} , {α,α})
$\rightarrow_{goat}$ { If the shepherd does not take the goat, then the result is the previous state. }
(∅, {shepherd,goat,α,α})

Derivation 1 The generic solution of the cabbage-goat-wolf problem

Note that by applying trivial rules to find this solution, we do not have to make any choices. Abstraction from irrelevant details resulted in the most general solution to the problem.

3. Strengthening the problem

The second technique we show is strengthening of the problem. We illustrate this technique by solving the two-married-couples problem.

3.1. The two-married-couples problem

The **two-married-couples problem**[2] is a variation of the cabbage-goat-wolf problem. In this problem two married couples have to cross a river. The couples have a boat at their disposal, but it can carry at most two people. The men of the couples do not trust each other. Therefore, no woman may be left alone (on a riverbank or in the boat) with the man of the other couple. How do the married couples cross the river?

3.2. The solution

We start solving this problem by introducing names: We give the two couples the names A and B. Further we introduce the names MA for the male of couple A, FA for the female of couple A, MB for the male of couple B, and FB for the female of couple B.

Now we have introduced names, we can characterise the forbidden states. Just as in Section 2.2, a **state** is a pair of sets: The first set element is the set of people on the left riverbank, the second element is the set of people on the right riverbank. In the forbidden states one of these sets equals {FA,MB}, {FA,MB,FB}, {FB,MA}, or {FB,MA,FA}.

The next step in our solution is strengthening the problem. We observe that in each forbidden state the males are *not* together. We exclude the forbidden states by always keeping the males together. This limitation reduces the number of possible solutions and it might even exclude all solutions!

The consequence of strengthening the problem is that we lose the distinction between the males and the distinction between the females. Because the males are always together, the problem is symmetric in the married pairs: In any state we can exchange FA and FB, and we can exchange MA and MB, without changing allowed states to forbidden states and *vice versa*. We therefore omit the A's and B's from the names of the males and females.

Now we can write down the solution to the problem. Just as in Section 2.2, we assume that the married pairs always keep the males together and never will do an action that brings them back in a state they already have been in. We use the same notation as in Section 2.2. The solution is shown in Derivation 2.

[2] Prof. Dr. E.W. Dijkstra says this problem can not be told in the USA because it is too sexist.

({M,M,F,F} , ∅)

$\rightarrow_{FF}$ { Moving F or MM results in returning to an already visited state. }

({M,M}, {F,F})

$\leftarrow_{F}$ { Moving back FF returns to an already visited state. }

({M,M,F} , {F})

$\rightarrow_{MM}$ { Moving F returns to an already visited state; keep males together. }

({F} , {M,M,F})

$\leftarrow_{F}$ { Moving back MM returns to an already visited state. }

({F,F} , {M,M})

$\rightarrow_{FF}$ { Moving F returns to an already visited state. }

(∅, {M,M,F,F})

Derivation 2 The generic solution of the two-married-couples problem

4. Application of simple theories

The last technique we show is application of simple theories. We illustrate this technique by solving the missionaries-cannibals problem.

4.1. The missionaries-cannibals problem

The missionaries-cannibals problem is another variation on the cabbage-goat-wolf problem. In this problem three missionaries and three cannibals have to cross a river. Again, they have a boat at their disposal that can carry at most two people. If at any moment more missionaries than cannibals are on the same riverbank, then the missionaries will convert the cannibals. How do the missionaries and the cannibals cross the river without any cannibal being converted?

4.2. The solution

We start solving this problem by proving a number of simple theories. These theories are about the distribution of the missionaries and the cannibals. We use the notation (*m*,*n*) for the set of **allowed** states in which *m* people are on one riverbank and *n* people are on the other riverbank. Note that (*m*,*n*) and (*n*,*m*) denote the same state.

Theory 1: In any state from the set (1,5) the single person is a missionary.
Proof: Assume the single person is not a missionary. Then the group of five consists of three missionaries and two cannibals. The two cannibals will be converted. This contradicts the assumption that the states from (1,5) are allowed states. Therefore the single person is a missionary.

Theory 2: In any state from the set (2,4) the group of two does not consist of two cannibals.
Proof: Assume the group of two consists of two cannibals. Then the group of four consists of three missionaries and one cannibal. This cannibal will be converted. This contradicts the assumption that the states from (2,4) are allowed states. Therefore the group of two does not consist of two cannibals.

Theory 3: In any state from the set (3,3) both triples are homogeneous (*i.e.* contain only missionaries or only cannibals).
Proof: Assume that the triples are not homogeneous. Then in one group must be two missionaries and one cannibal, and in the other group must be one missionary and two cannibals. The cannibal in the first group will be converted. This contradicts the assumption that the states from (3,3) are allowed states. Therefore both triples are homogeneous.

Now we can write down the solution of the problem. We denote a missionary by an ‘m’, and we denote a cannibal by a ‘c’. Just as in Section 2.2, we assume that the missionaries and cannibals never will do an action that brings them back in a state they already have

been in. Further, we always send two people from the left riverbank to the right riverbank, because this results in the most efficient solution. We use the same notation as in Section 2.2. The solution is shown in Derivation 3.

({c,c,c,m,m,m} , ∅)
$\rightarrow_{mx}$ { Sending two persons results in a state from (2,4). Theory 2 implies that we must send at least one missionary. Let 'x' be an 'm' or a 'c'. }
({c,c,c,m,m} – {x} , {m,x})
$\leftarrow_{x}$ { Sending back mx returns to an already visited state. Returning 'm' leads to conversion of cannibals if x = c. }
({c,c,c,m,m} , {m})
$\rightarrow_{mm}$ { Sending two persons results in a state from (3,3). Theory 3 implies that we must send two missionaries. }
({c,c,c} , {m,m,m})
$\leftarrow_{m}$ { Sending back two missionaries results in an already visited state. }
({m,c,c,c} , {m,m})
$\rightarrow_{cc}$ { Sending two persons results in a state from (2,4). Theory 2 implies that we must send two cannibals. Sending one person leads to (3,3) which implies we must send the missionary. This leads to an already visited state. }
({m,c} , {m,m,c,c})
$\leftarrow_{mc}$ { Sending two persons results in a state from (2,4). Theory 2 implies that we must send one cannibal and one missionary. Sending one person results in two heterogeneous groups of 3 persons, which is forbidden by Theory 3. }
({m,m,c,c} , {m,c})
$\rightarrow_{cc}$ { The previous step is symmetric: (V,W) $\rightarrow_{cc}$ (W,V)). Therefore the rest of the solution is the reverse of the first part of the solution.[3] }
({m,m} , {m,c,c,c})
$\leftarrow_{m}$ { }
({m,m,m} , {c,c,c})
$\rightarrow_{mm}$ { }
({m} , {m,m,c,c,c})
$\leftarrow_{x}$ { }
({m,x} , {m,m,c,c,c} – {x})
$\rightarrow_{mx}$ { }
(∅ , {m,m,m,c,c,c})

Derivation 3 The generic solution of the missionaries-cannibals problem

[3] This holds also for the examples of the previous sections.

5. Applying the techniques to another example

This chapter shows how the techniques introduced in the previous chapters help us solving another problem. This problem is not an example given by Prof. Dr. Dijkstra, but it was the final assignment for a course on formal methods. After having solved the problem we proved its correctness using the tool SPIN. (SPIN is a tool to prove the correctness of assertions in programs.)

5.1. The flipping-glasses problem

Imagine a square platform on top of a spindle. This platform can be rotated over 90, 180, 270, or 360 degrees. At each corner of the platform stands a glass. Each glass can stand in its normal position or upside down. Glasses are specified by their location on the platform, as shown in Figure 4. A **location** is a number from 1 to 4. A rotation changes the location of glasses. For example, after a rotation over 90 degrees the glass at location 1 has moved to location 2, and the glass at location 4 has moved to location 1. The arrow in this figure denotes the rotation direction of the platform.

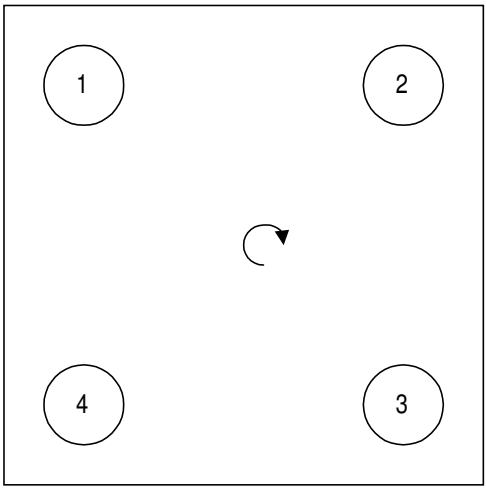

Figure 4 The platform and the four positions of the glasses

Two players, A and B, are playing a game. Player B is blindfolded so that he can not see the platform and the glasses. Initially, at least one of the glasses is upside down. The game consists of a number of turns. Each **turn** consists of the following steps:

1. Player A rotates the platform over 90, 180, 270, or 360 degrees.
2. Player B tells player A for each location whether the glass must be flipped. At least one glass must be flipped.
3. Player A flips the glasses. If all glasses are now in their normal position the game ends. Player A informs player B whether the game has ended.

Player B wins the game if the game ends. Player A wins the game if player B gives up. The problem is to find a winning strategy for player B.

5.2. The solution

At first sight the game seems hopeless for player B, since he has no information about the glasses. Therefore we have to develop a method to get this information. Here is an example of how B can get this information: If player B lets flip all four glasses and the game ends, then in the initial state all glasses were upside down. If the game does not end, then still at least one of the glasses is upside down.

Because player B does not know how player A rotates the platform, we have to find a solution that is not influenced by platform rotations.

We introduce configurations and classes of configurations. A **configuration** is a quadruple (g_1, g_2, g_3, g_4), where for all n, $n \in [1..4]$, g_n is a boolean expression with the value "the glass on location n stands in the normal position." A **class** of configurations is a set of configurations that is closed under rotation of the table and closed under flipping all four glasses. This means that if c is a configuration of class C, then all rotations of c are also configurations of class C. Moreover, flipping the four glasses of configuration c delivers a configuration of C.

It is clear that for all n, $n \in \{0, 1, 2\}$, the set C_n is a class, where C_n is defined as $C_n = \{ c \mid c$ is a configuration in which either n or $4\text{-}n$ glasses are in normal position $\}$. Figure 5 shows a graph representing the possible transitions between these classes. The nodes are the classes. An edge between node C_m and C_n indicates that there exists a configuration c, $c \in C_m$, that can be changed into a state of C_n by flipping the number of glasses specified in the label of the edge. Flipping 4 does not change the class of a configuration. Therefore we omit the self-loops with label "4".

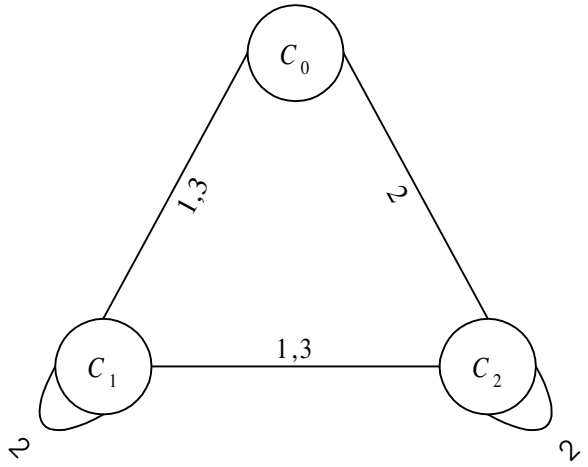

Figure 5 Transitions between the classes C_0, C_1, and C_2

The game starts in one of the three states mentioned in Figure 5. If the game starts in state C_1 or C_2, and player A knows which glasses player B wants to flip, then player A can prevent player B from winning the game. Player A can rotate the platform each turn in such a way that the configuration of the game stays in classes C_1 or C_2.

We can not solve the problem, because we abstracted too much. The class C_2 consists actually of two classes: C_{2d} and C_{2nd}, which are defined as
C_{2d} = { c | c is a configuration in which 2 diagonal glasses are in normal position }, and
C_{2nd} = { c | c is a configuration in which 2 non-diagonal glasses are in normal position }.
Diagonal glasses are two glasses whose locations have a difference of two.

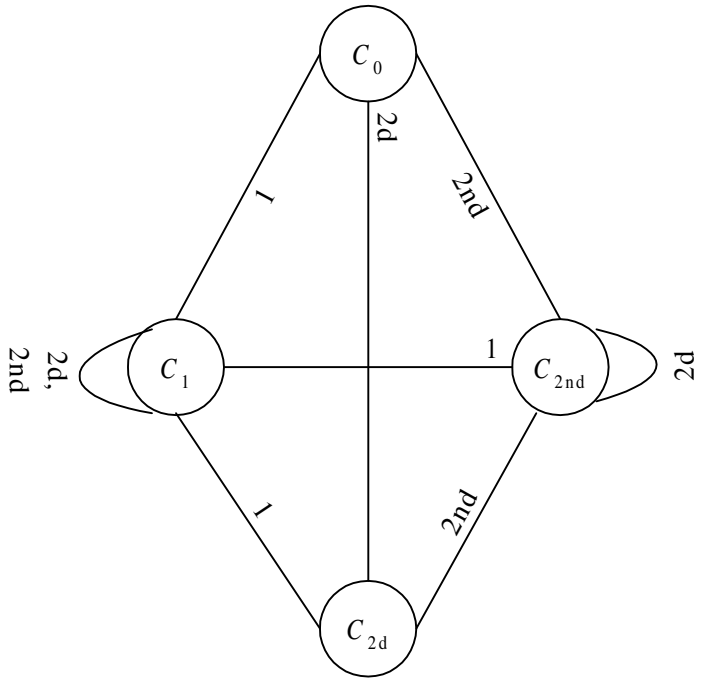

Figure 6 Transitions between the classes C_0, C_1, C_{2d}, and C_{2nd}

Figure 6 shows the transitions between the new classes. The label "2d" stands for flipping two diagonal glasses; the label "2nd" stands for flipping two non-diagonal glasses.

We can finally derive a solution to the problem. The derivation is a list of states. A **state** is a set of configurations. If the game has not yet ended, then the configuration of the platform is an element of this set. In a derivation the states are separated by transitions. A **transition** is denoted by the word "flip" followed by the number of glasses that player B wants to be flipped. The transition "flip 2d" represents flipping two diagonal glasses; "flip 2nd" represents flipping two non-diagonal glasses.

Again, we forbid transitions that result in an already visited state. The result is that for each transition we have only one choice. The result is shown in Derivation 4.

	$C_0 \cup C_1 \cup C_{2d} \cup C_{2nd}$
flip 4	{ }
	$C_1 \cup C_{2d} \cup C_{2nd}$
flip 2d	{ }
	$C_1 \cup C_0 \cup C_{2nd}$
flip 4	{ We introduced C_0 again. We remove it in the same way as before.}
	$C_1 \cup C_{2nd}$
flip 2nd	{ We introduced C_{2d} and C_0 again. We remove them in the same way as before. }
	$C_1 \cup C_{2d} \cup C_0$
flip 4	{ }
	$C_1 \cup C_{2d}$
flip 2d	{ }
	$C_1 \cup C_0$
flip 4	{ }
	C_1
flip 1	{ }
	$C_0 \cup C_{2d} \cup C_{2nd}$
flip 4	{ We introduced C_{2nd}, C_{2d} and C_0 again. We remove them in the same way as before. }
	$C_{2d} \cup C_{2nd}$
flip 2d	{ }
	$C_0 \cup C_{2nd}$
flip 4	{ }
	C_{2nd}
flip 2nd	{ }
	$C_{2d} \cup C_0$
flip 4	{ }
	C_{2d}
flip 2d	{ }
	C_0
flip 4	{ }
	$\varnothing$

Derivation 4 The generic solution of the flipping-glasses problem

6. Conclusions

In his presentation Prof. Dr. E.W. Dijkstra showed three techniques to reduce the number of case analyses, while searching for a solution to a problem: abstraction from irrelevant details, strengthening the problem, and application of simple theories. These techniques are illustrated by solving three simple problems.

In the first problem, the cabbage-goat-wolf problem, we have abstracted from the "direction" of eating. It does not matter whether the wolf eats the goat or the goat eats the wolf. This has lead to a unique and generic solution.

In the second problem, the two-married-couples problem, we have concluded that in the forbidden states the males are not at the same riverbank. We have strengthened the problem by keeping the males always together. Again, this has lead to a unique and generic solution.

In the third problem, the missionaries-cannibals problem, we have proved three simple theories. By applying these theories we have found a unique and generic solution again.

In the last problem, the flipping-glasses problem, we have abstracted from the precise configuration of the platform. Only the number of glasses standing in the normal position is relevant, not their locations. Again, we have found a unique and generic solution.

The examples show that application of the discussed techniques indeed reduces the number of case analyses. If we omit choices of which we can see after one transition that they do not lead to a solution, then application of the techniques delivers us a unique and generic solution.