A Procedural Solution to the Unexpected Hanging and Sorites Paradoxes*

Stuart C. Shapiro
Department of Computer Science
And Center for Cognitive Science
State University of New York at Buffalo
226 Bell Hall
Buffalo, New York 14260-2000
U.S.A.
phone: 716-645-3180 Ext. 125
fax: 716-645-3464
e-mail: shapiro@cs.buffalo.edu

March 9, 1998

Abstract

The paradox of the Unexpected Hanging, related prediction paradoxes, and the sorites paradoxes all involve reasoning about ordered collections of entities: days ordered by date in the case of the Unexpected Hanging; men ordered by the number of hairs on their heads in the case of the bald man version of the sorites. The reasoning then assigns each entity a value that depends on the previously assigned value of one of the neighboring entities. The final result is paradoxical because it conflicts with the obviously correct, commonsensical value. The paradox is due to the serial procedure of assigning a value based on the newly assigned value of the neighbor. An alternative procedure is to assign each value based only on the original values of neighbors—a parallel procedure. That procedure does not give paradoxical answers.

1 The Paradox

A version of the well-known Unexpected Hanging Paradox is:

> Having been convicted of murder, Mr. Hyde hears the judge's sentence on Thursday, April 28: "One morning next week, you will be taken out and hanged by the neck until dead. Moreover, you will not know which morning this will happen until the warden comes to your cell at dawn to take you to the gallows." Back at his cell, Mr. Hyde reasons as follows: "If I am alive at noon on Friday, I will know that I will be hanged on Saturday. Thus, I will know when I will be hanged before the warden comes on Saturday, so I can't be hanged on Saturday, May 7. But therefore, if I am alive at noon on Thursday, I will know that I will be hanged on Friday, so I can't be hanged on Friday, May 6. Similarly, I can't be hanged on Thursday, Wednesday, Tuesday, Monday, or Sunday, May 1. So the judge can't carry out his sentence, and I can relax. I won't be hanged." Mr. Hyde then relaxed, and was, indeed, surprised when the warden came for him at dawn on Wednesday, May 4, and hanged him.

*This is a preliminary version of Shapiro, Stuart C. A Procedural Solution to the Unexpected Hanging and Sorites Paradoxes. *Mind,* 1998. All quotes should be from, and all citations should be to the published version.

The paradox is that Mr. Hyde's reasoning seems impeccable, yet the judge's sentence was carried out, and Mr. Hyde was, indeed, surprised and hanged. How can we reconcile this? The answer given in this paper is that there are three ways to reason about the first week of May. Mr. Hyde reasoned serially from Saturday, May 7 back to Sunday, May 1 and concluded thereby that he couldn't be hanged. There are, however, two other ways of reasoning: serially from Sunday, May 1 forward to Saturday, May 7; or in parallel, reasoning about all days at once. Either of these latter ways of reasoning finds days on which the judge can properly hang Mr. Hyde. The paradox is due to the clash of different answers found by the different reasoning methods. The fact that Hyde's answer wasn't correct is due to the fact that he used an inappropriate method of reasoning.

2 The Literature

The Unexpected Hanging has been discussed in several forms in a number of papers and books starting with (O'Connor, 1948). Good discussions and bibliographies are in (Hughes and Brecht, 1978), (Gardner, 1991), (Sainsbury, 1995), and (Chow, 1998). (Gardner, 1991) has a bibliography of 57 items dating from 1948 to 1988. A recent opinion on the paradox is that Hyde's reasoning shows the judge's sentence to be self-contradictory. As far as I know, the procedural solution given in this paper has not previously been suggested.

3 Procedures and Correctness

In this paper, I take a *procedure* to be an unambiguously specified set of instructions for solving a particular class of problems. For example, one might specify a procedure for adding sequences of two-digit base ten integers. The specific problem of adding `58`, `15`, and `73` is an instance of this class of problems.

There are two notions of "correctness" applied to procedures that are relevant to this paper. The first is whether or not a procedure is correctly followed. This requires that the procedure is unambiguous for the person or machine attempting to carry it out, and that that person or machine does not make a mistake in doing so. In this paper, I assume that all procedures are followed correctly. I am not arguing that any of the paradoxes discussed in this paper depend on incorrectly following a procedure.

The second notion is whether or not a procedure is correct (or *appropriate*) for a particular class of problems. Determining this can be difficult. For some classes of problems, procedures can be formally derived from the formal specifications of the problem (Gries, 1981), and for some classes of problems, procedures can be formally proven to be correct (Dijkstra, 1976). However, in most cases, the best we can do is test a procedure against problem instances for which we know the correct answers. If the procedure produces answers we know are correct, we gain some confidence that the procedure may be correct. However, if the procedure produces answers we know are incorrect, we know that the procedure is incorrect (that is, inappropriate for the given class of problems), and we either reject that procedure or try to debug it. My analysis of the paradoxes in this paper is that each one depends on following a decision procedure that produces a paradoxical answer. The answer is paradoxical because it conflicts with an answer we know to be correct. My solution is that this is a reason for rejecting the procedure as inappropriate. In each case, I present an alternate, appropriate procedure that produces the correct answer.

4 Serial and Parallel Procedures

The rise of parallel computing has familiarized many people with the distinction between serial and parallel procedures. Although any parallel procedure may be simulated by a serial procedure, straightforward serial and parallel approaches to a single problem may yield different answers. For example, consider the problem of interchanging the values of two variables in a computer program. Attempting this serially, without any extra variable, fails regardless of the order in which we copy the value of one variable to the other. Assume the variable `x` starts off with the value `1` and `y` starts off with the value `2`. The serial procedure

```
Set x to the value of y;
Set y to the value of x.
```

leaves both variables with the value 2, while the serial procedure

```
Set y to the value of x;
Set x to the value of y.
```

leaves both variables with the value 1. However, if we set x to y and y to x in parallel—at exactly the same time—we would succeed in leaving x with the value 2 and y with the value 1. Note that the first procedure, correctly followed, sets both variables to 2; the second procedure, correctly followed, sets both variables to 1; and the third procedure, correctly followed, sets x to 2 and y to 1. The three answers are all correctly computed from the three procedures, but only the parallel procedure accomplishes what was desired in the original statement of the problem, so only that prodecure is appropriate for this problem.

For another example, consider a vector [a_1, ..., a_7] of numbers. We want to change each element of the array to the average of itself and its two neighbors. Of course, the two ends don't have two neighbors, so we will change each of them to the average of itself and its one neighbor. Assume the vector starts off with the values

[30, 6, 66, 36, 12, 18, 0].

After the parallel procedure

```
for i from 1 to 7 in parallel do
    if i=1 then set a_i to (a_i + a_{i+1})/2
    else if i=7 then set a_i to (a_{i-1} + a_i)/2
    else set a_i to (a_{i-1} + a_i + a_{i+1})/3
```

the values will be [18, 34, 36, 38, 22, 10, 9]. However, after the serial procedure

```
for i from 1 to 7 in serial do
    if i=1 then set a_i to (a_i + a_{i+1})/2
    else if i=7 then set a_i to (a_{i-1} + a_i)/2
    else set a_i to (a_{i-1} + a_i + a_{i+1})/3
```

the values will be [18, 30, 44, 30.7, 20.2, 12.7, 6.4], because each element of the vector is changed before it is used to compute the next element. If we run the serial procedure in the other direction,

```
for i from 7 to 1 in serial do
    if i=1 then set a_i to (a_i + a_{i+1})/2
    else if i=7 then set a_i to (a_{i-1} + a_i)/2
    else set a_i to (a_{i-1} + a_i + a_{i+1})/3
```

the values will be [27.3, 24.5, 37.6, 40.8, 20.3, 13, 9], because this time, it is the right neighbor that is changed before each element is computed. Again, notice that [18, 30, 44, 30.7, 20.2, 12.7, 6.4] is the correct answer for the forward serial procedure, and [27.3, 24.5, 37.6, 40.8, 20.3, 13, 9] is the correct answer for the backward serial procedure, but neither is an appropriate procedures to use if we understand the statement of the problem to require that each a_i be changed to the average of it and its neighbors using the neighbors' original values.

For an example for which the forward serial procedure is the appropriate procedure to use and the parallel and backward serial procedures are inappropriate, consider creating a table of factorials. The factorial of an integer i is the product of all integers from 1 through i. We will do this by first initializing a vector [$fact_0$, ..., $fact_{max}$] so that $fact_0$ = 1 and $fact_i$ = i, for $1 \leq i \leq max$. Then we will use either the forward serial procedure

```
for i from 1 to max in serial do
    set fact_i to fact_i * fact_{i-1}.
```

the backward serial procedure

```
for i from max to 1 in serial do
    set fact_i to fact_i * fact_i-1.
```

or the parallel procedure

```
for i from 1 to max in parallel do
    set fact_i to fact_i * fact_i-1.
```

The following table shows the initialized vector and the results of the three procedures for `max = 10`.

i:	0	1	2	3	4	5	6	7	8	9	10
initial vector	1	1	2	3	4	5	6	7	8	9	10
after forward serial	1	1	2	6	24	120	720	5040	40320	362880	3628800
after backward serial	1	1	2	6	12	20	30	42	56	72	90
after parallel	1	1	2	6	12	20	30	42	56	72	90

Each of the three answers is the correct answer for its procedure, but only the forward serial procedure produces the correct answer for the stated problem.

5 Serial and Parallel Procedures Applied to the Unexpected Hanging

We can reformulate the above version of the Unexpected Hanging as follows. Consider a vector `[May 1, ..., May 7]` of 7 truth values, initially all `U` (unknown). Assign an element the value `F` if either it is the last element in the vector or if every element after it has the value `F`, otherwise assign it the value `T`. Any element that has the value `T` after this procedure represents a day on which Hyde can be unexpectedly hanged. Hyde used the following serial procedure.

```
for i from 7 to 1 in serial do
    if i=7 or if May i+1 through May 7 are all F
       then set May i to F
    else set May i to T.
```

It is easy to see that at the end of this procedure, all the elements have the value `F`, and there is no element with the value `T`, so this procedure says that Hyde can't be unexpectedly hanged.

There are, however, at least two other procedures that could be used: the parallel procedure,

```
for i from 1 to 7 in parallel do
    if i=7 or if May i+1 through May 7 are all F
       then set May i to F
    else set May i to T.
```

and the forward serial procedure,

```
for i from 1 to 7 in serial do
    if i=7 or if May i+1 through May 7 are all F
       then set May i to F
    else set May i to T.
```

After either of these only `May 7` has the value `F`; `May 1` through `May 6` all have the value `T`, meaning that Hyde can be surprised and hanged on any day before Saturday, May 7.

As above, Hyde's reasoning is correct in the sense that he correctly got the answer computed by his procedure. However, he used an inappropriate procedure for the problem. The way to see that Hyde's procedure is inappropriate is to note that it got the wrong answer. As Scriven says (Scriven, 1951, p. 403) "the monster, Reality" proves him wrong. How do we know that Hyde's answer is incorrect? That is something I do not feel I have to explain. Everyone who has ever discussed this paradox agrees that

Hyde's answer is wrong. That is why this is considered a paradox. We all seem to decide that Hyde can be surprised and hanged by using, perhaps implicitly, either the parallel or the forward serial procedure. If Hyde had been more clever, he could have done the same. Alternatively, we may note that Hyde's reasoning assumes both the truth of the judge's prediction and the appropriateness of Hyde's reasoning procedure. The accuracy of the judge's statement may be preserved by taking Hyde's *reductio* argument to contradict the appropriatness of his own reasoning procedure. The key is realizing that there are alternative reasoning procedures.

5.1 Non-linearly Ordered Versions

Sorensen (Sorensen, 1982) shows that the Unexpected Hanging and similar paradoxes do not rely on a strict, linear ordering of the entities being examined. However, he does order them by groups. For example, in his "undiscoverable position paradox," the entities are the nine positions of a 3 by 3 matrix. One ordering he considers is first the four corners, then the four sides, and finally the center. Although he considers other orders as well, all his paradoxical reasoning procedures consider the positions in some order, with the property of each position depending on the property of positions in the earlier groups. The answer he considers to be correct is achieved by considering the positions in parallel. Sorensen's other "recalcitrant variations" also fit the analysis presented in this present paper.

6 Sorites Paradoxes

A set of paradoxes usually considered to be unrelated to the Unexpected Hanging and its cousins are the sorites, or heap, paradoxes (see (Sainsbury, 1995, pp. 23–51)). A standard example of this paradox is:

> A man with no hair on his head is certainly bald. If a man with 0 hairs is bald, then so is a man with 1 hair. If a man with 1 hair is bald, then so is a man with 2 hairs. ...Therefore, all men are bald regardless of the number of hairs on their heads. However, this is certainly false, since a man with a full head of hair is not bald.

These paradoxes are usually laid at the feet of vagueness. Sainsbury says,

> "As with any paradox there are three possible responses to consider:
>
> (a) Accept the conclusion of the argument.
>
> (b) Reject the reasoning as faulty.
>
> (c) Reject one or more premises.
>
> ...Sorites reasoning appears to be extremely simple and to use only the fundamental logical principle of modus ponens, so the prospect of rejecting it as invalid (response (b) above) is not initially tempting." [(Sainsbury, 1995, pp. 30–31)]

When he returns to consider response (b), the discussion is in terms of degrees of truth—vagueness.

I have no disagreement with the notion that baldness (and its cousin concepts in other versions of the sorites) is a vague notion in the sense that there is no clean dividing line between bald and non-bald. That is, there is no number θ that we can all agree on, such that men with fewer than θ hairs are bald and men with at least θ hairs are not bald. In fact, there is probably no such θ that any one person uses consistently to separate the bald from the non-bald.

Nevertheless, there is still a problem. Let `DecideBaldness(n)` be a procedure for deciding whether a man with `n` hairs is bald. I do not care how `DecideBaldness` works—with a fixed θ, probabilistically, nondeterministically, randomly—as long as `DecideBaldness(0)` returns `B`, for bald, and `DecideBaldness(maxhairs)` returns `N`, for not bald, (where `maxhairs` is the number of hairs a man with a full head of hair has). The problem is that the procedure given above still calls all men bald, so we still have a paradox.

I will use Sainsbury's response (b), but instead of blaming modus ponens or ending the analysis with the notion of vagueness, I will appeal to the difference between serial and parallel decision procedures.

Let $\mathtt{man_i}$ represent a man with `i` hairs on his head; $\mathtt{man_i}$ will have the value `B` if $\mathtt{man_i}$ is bald, `N` if he is not bald, and `?` if we have not yet decided whether he is bald. I will assume that if we see a man with

i hairs on his head in isolation, then, we will consider him to be bald or not according to the procedure `DecideBaldness(`i`)`. However, if we have already decided that $\texttt{man}_{\texttt{i-1}}$ is bald, then we will consider $\texttt{man}_{\texttt{i}}$ to be bald. Similarly, if we have already decided that $\texttt{man}_{\texttt{i+1}}$ is not bald, then we will consider $\texttt{man}_{\texttt{i}}$ not to be bald. Assume the most hairs on any man's head is `maxhairs`. We could use three different procedures to decide baldness for any number of hairs—the forward serial procedure,

```
for i from 0 to maxhairs in serial do
    if i>0 and man_{i-1} has the value B
       then set man_i to B
    else if i<maxhairs and man_{i+1} has the value N
       then set man_i to N
    else set man_i to DecideBaldness(i).
```

the backward serial procedure,

```
for i from maxhairs to 0 in serial do
    if i>0 and man_{i-1} has the value B
       then set man_i to B
    else if i<maxhairs and man_{i+1} has the value N
       then set man_i to N
    else set man_i to DecideBaldness(i).
```

or the parallel procedure,

```
for i from 0 to maxhairs in parallel do
    if i>0 and man_{i-1} has the value B
       then set man_i to B
    else if i<maxhairs and man_{i+1} has the value N
       then set man_i to N
    else set man_i to DecideBaldness(i).
```

The forward serial procedure will decide that all men are bald, the backward serial procedure will decide that all men are not bald, and the parallel procedure will decide each man independently by the procedure `DecideBaldness`. The paradox is resolved, because the procedure that we actually use in real life is the parallel procedure, whereas the procedure used at the beginning of this section, and whenever the sorites paradoxes are presented, is a serial procedure.

6.1 A Randomly Ordered Procedure

If you don't like the idea of considering all men in order of the number of hairs on their head, or all at once, we could, instead, use a randomly ordered prodecure. To make this simpler, we will use two dummy men: $\texttt{man}_{\texttt{-1}}$ with an initial value of `B`; and $\texttt{man}_{\texttt{maxhairs+1}}$ with an initial value of `N`.

```
for i from 0 to maxhairs randomly do
    if man_{i-1} has the value ?  and man_{i+1} has the value ?
       then set man_i to DecideBaldness(i)
    else if man_{i-1} has the value ?
       then set man_i to the value of man_{i+1}
    else if man_{i+1} has the value ?
       then set man_i to the value of man_{i-1}
    else if man_{i-1} = man_{i+1}
       then set man_i to the value of man_{i-1}
    else set man_i to DecideBaldness(i).
```

This procedure considers each $\texttt{man}_{\texttt{i}}$ in random order of `i`. If we have not yet decided about the baldness of either of the two neighboring men, $\texttt{man}_{\texttt{i-1}}$ or $\texttt{man}_{\texttt{i+1}}$, or if these two men have different values of baldness, we will decide about the baldness of $\texttt{man}_{\texttt{i}}$ according to `DecideBaldness(i)`. However, if we have decided the baldness of exactly one of the two neighbors, or if both neighbors have the same value, then we give that value to $\texttt{man}_{\texttt{i}}$.

To see how this works, I show in the following table the results of running this procedure 10 times using 20 as the value of `maxhairs`, and having `DecideBaldness(i)` return the value of `B` with probability `(maxhairs - i)/maxhairs`.

run	man																				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	B	B	B	B	B	B	B	B	B	N	N	B	B	N	N	N	N	N	N	N	N
2	B	B	B	B	N	N	B	B	B	B	B	B	N	N	N	B	B	N	N	N	N
3	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	N	N	N	N
4	B	B	B	B	B	B	N	N	N	N	B	B	B	N	N	N	N	N	N	N	N
5	B	B	B	B	B	B	B	B	B	B	B	B	N	N	N	N	N	N	N	N	N
6	B	B	B	B	B	B	B	N	N	N	N	N	N	N	N	N	N	N	N	N	N
7	B	B	N	N	N	N	N	N	B	B	B	B	N	N	N	N	N	N	N	N	N
8	B	B	B	B	B	B	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
9	B	B	B	B	B	B	N	N	N	B	B	B	B	N	N	N	N	N	N	N	N
10	B	B	B	B	B	B	B	B	B	B	B	B	B	N	N	N	N	N	N	N	N

7 Conclusion

The Unexpected Hanging, related prediction paradoxes, and the Sorites paradoxes have seemed to be paradoxes because apparently impeccable reasoning leads to one conclusion, whereas commonsense, or Reality, shows a contradictory conclusion to be correct. All these paradoxes involve reasoning about ordered collections of entities. The paradoxical reasoning procedures assign each entity a value that depends on previously assigned values of neighboring entities. Nonparadoxical conclusions can be reached by using a parallel procedure that uses the original values of the neighboring entities, or by using a randomized procedure, or a serial procedure that considers the entities in a different order. Although these paradoxes are caused by incorrect reasoning, it is not the reasoning rules of standard logic that are at fault, but the procedures used to select the order in which entities are chosen to which to apply those rules.

Acknowledgments

This paper was inspired by the paper, "Treatment of Conflict: The Pragmatic Dimension of Paradox," given by Mariam Thalos to the University at Buffalo Center for Cognitive Science. I thank her for that inspiration and for her comments on an earlier draft of this paper. I also thank William J. Rapaport for helpful comments on earlier drafts and for his encouragement to write this paper in the first place. Thanks also to other members of the SNePS Research Group, namely Debra T. Burhans, Alistair E. Campbell, Haythem O. Ismail, Geoffrey D. Koplas, Frances L. Johnson, and Bruce Wisenburn who also gave useful comments on an earlier version of this paper.

References

Chow, T. Y. (1998). The surprise examination or unexpected hanging paradox. *The American Mathematical Monthly*, 105(1):41–51.

Dijkstra, E. W. (1976). *A Discipline of Programming*. Prentice-Hall, Englewood Cliffs, NJ.

Gardner, M. (1991). *The Unexpected Hanging and Other Mathematical Diversions*. The University of Chicago Press, Chicago, IL.

Gries, D. (1981). *The Science of Programming*. Springer-Verlag, New York.

Hughes, P. and Brecht, G. (1978). *Vicious Circles and Infinity*. Penguin Books, New York.

O'Connor, D. J. (1948). Pragmatic paradoxes. *Mind*, 57(227):358–359.

Sainsbury, R. M. (1995). *Paradoxes*. Cambridge University Press, Cambridge, UK, second edition.

Scriven, M. (1951). Paradoxical announcements. *Mind*, 60(239):403–407.

Sorensen, R. A. (1982). Recalcitrant variations of the prediction paradox. *Australasian Journal of Philosophy*, 69(4):355–362.