NOTE:

An Exercise in Weakest Preconditions

Robin Whitty

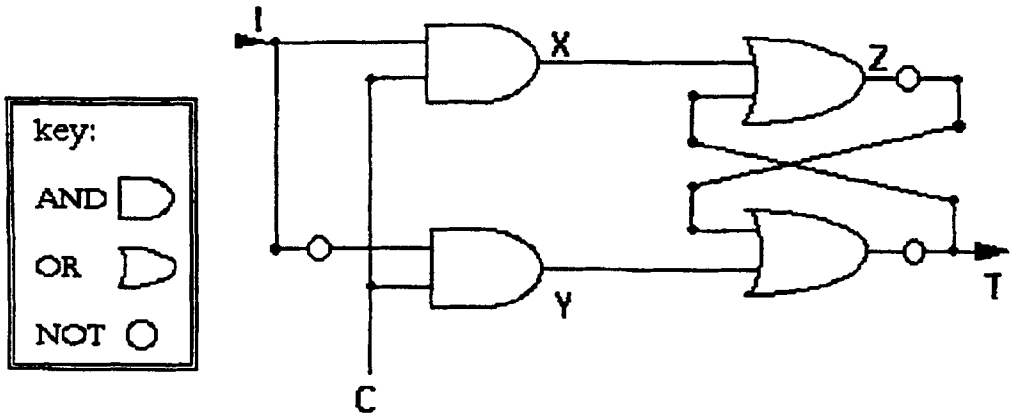
Centre for Systems and Software Engineering, Department of Computing and Mathematics, South Bank Polytechnic, Borough Road, London, SE1 0AA, UK

Abstract

Weakest preconditions are used to formulate the requirements for a 2-state memory cell and it is proved that the flip-flop device meets these requirements. This is an exercise in the use of the weakest preconditions which is more realistic than the usual examples of non-looping arithmetic algorithms.

1.0 Introduction

Undergraduate Computing Science students are frequently mystified on first being shown the construction of a 2-state memory cell using logic gates AND, OR, and NOT.



The action of the device is explained in terms of the propagation of the input I and control C through X, Y and Z to give output T. A neat way of representing this propagation is the following Pascal procedure:

```
PROCEDURE Flip_Flop (C, I : BOOLEAN; VAR T : BOOLEAN);
VAR X, Y, Z : BOOLEAN;
BEGIN
S1:   X :=0; Y:=0; Z :=1; {these are default values: X,Y and Z may be
                                redefined during S2-S4}
S2:   IF C THEN
            IF I THEN X:=1
            ELSE Y:=1;
S3:   IF NOT X THEN
            IF NOT T THEN Z :=0;
S4:   IF NOT Y THEN
            IF Z THEN T:=1
            ELSE T :=0
      ELSE T :=0
END;
```

S1 - S4 are used to denote the separate statements within the procedure body.

One aspect of the flip-flop which causes confusion is that, while C and I are input parameters to the procedure which are known in advance, it is not clear what initial value T will take. This can be proved formally to be unimportant, as we shall see.

At some stage, it is likely that students on a modern Computing Science course will learn how to use weakest preconditions in the design of provably correct programs. In this context it is instructive to refer again to the flip-flop and ask how weakest preconditions can be used to formulate the requirements for a 2-state memory cell, and to prove that the flip-flop meets these requirements. This is an exercise in the use of weakest preconditions which is more realistic than the usual examples of non-looping arithmetic algorithms. It can be introduced after the rules for sequential and conditional composition have been explained and gives students some thorough practice in these rules before tackling the much more demanding concepts of weakest preconditions for loops and loop invariants. It recalls and reinforces some standard manipulation techniques from propositional calculus, which the students will have met already. It also gives a valuable lesson in the use of good, concise notation and a structured approach to problem solving - things can easily get out of hand if these principles are disregarded during the exercise.

1.1 Specification of Requirements

The most basic way to capture the required behaviour of the flip-flop is to describe this behaviour for two successive pairs of signals C_0, I_0 and C_1, I_1. This behaviour is modelled by two calls to our Pascal procedure:

S = Flip_Flop(C_0, I_0, T); Flip_Flop(C_1, I_1, T).

The postcondition says: if C_1 is 'off' (zero) then remember the previous input I_0 , otherwise remember the new input I_1:

$$Q = (\neg C_1 \Rightarrow T=I_0) \wedge (C_1 \Rightarrow T=I_1). \quad (1)$$

A precondition for S is now needed: (1) is only guaranteed to work if C_0 is 'on', otherwise C_{-1} may be being remembered:

$$P \equiv C_0. \quad (2)$$

1.2 Verification of the Implementation

The proposed implementation is the Pascal code for S give in section 1.0. The aim is to show that the specification, embodied in the pre- and postconditions P and Q in equations (1) and (2), is satisfied by S. Thus, it must be shown that {P} S {Q}. Since termination of S is certainly guaranteed, it is only necessary to show that

$$P \Rightarrow WP(S, Q).$$

Calculating this weakest precondition requires application of the sequential composition rule to S:

$$WP(S, Q) \equiv WP(Flip\_Flop(C_0, I_0, T), WP(Flip\_Flop(C_1, I_1, T), Q)).$$

Rather than mechanically working backwards through this calculation, attempts to take short cuts will be made wherever possible. Let

$$R \equiv WP(Flip\_Flop(C_1, I_1, T), Q).$$

The weakest precondition with an un-subscripted C and I will be calculated and expression (1) for Q will be substituted as late as possible. Then, the actual value of R uses the values $C=C_1$ and $I=I_1$, and WP(S,Q) will be an expression of the same form as R but using the values $C=C_0$ and $I=I_0$. Now, applying the sequential composition rule to R:

$$R \equiv WP(S_1, WP(S_2, WP(S_3, WP(S_4, Q)))).$$

Again time is saved by evaluating this from 'the outside in', leaving $WP(S_4, Q)$ till last. S_1 is just a simple sequence of applications of the assignment axiom, giving:

$$R \equiv WP(S_2, WP(S_3, WP(S_4, Q)))_{X=0,\ Y=0,\ Z=1}. \quad (3)$$

Applying the conditional composition rule to S_2 in (3) gives

$$\begin{aligned} R \equiv \neg C \vee [&(\neg I \vee WP(S_3, WP(S_4, Q))_{X=1,\ Y=0,\ Z=1}) \\ &\wedge (I \vee WP(S_3, WP(S_4, Q))_{X=0,\ Y=1,\ Z=1})] \\ &\wedge C \vee WP(S_3, WP(S_4, Q))_{X=0,Y=0,\ Z=1}. \end{aligned} \quad (4)$$

Applying the conditional composition rule for each occurrence of S_3 in (4), and substituting for the value of X gives:

$$\begin{aligned} R \equiv \neg C \vee [&(\neg I \vee WP(S_4, Q)_{Y=0, Z=1}) \\ &\wedge (I \vee \{T \vee WP(S_4, Q)_{Y=1, Z=0}\} \\ &\wedge \{\neg T \vee WP(S_4, Q)_{Y=1, Z=1}\})] \\ &\wedge C \vee [\{ T \vee WP(S_4, Q)_{Y=0, Z=0}\} \\ &\wedge \{\neg T \vee WP(S_4, Q)_{Y=0, Z=1}\}]. \end{aligned} \quad (5)$$

Applying the conditional composition rule for each occurrence of S_4 in (5), and substituting for the values of Y and Z gives:

$$\begin{aligned} R \equiv \neg C \vee [&(\neg I \vee Q_{T=1}) \\ &\wedge (I \vee \{T \vee Q_{T=0}\} \wedge \{\neg T \vee Q_{T=0}\})] \\ &\wedge C \vee [\{T \vee Q_{T=0}\} \wedge \{\neg T \vee Q_{T=1}\}]. \end{aligned} \quad (6)$$

Simplifying (6):

$$\begin{aligned} R \equiv \neg C \vee [&(\neg I \vee Q_{T=1}) \wedge (I \vee Q_{T=0})] \\ &\wedge C \vee [(T \vee Q_{T=0}) \wedge (\neg T \vee Q_{T=1})]. \end{aligned} \quad (7)$$

The actual value of R is obtained by making the substitutions $C=C_1$ and $I=I_1$ in (7). The overall solution is

$$WP(S, Q) \equiv WP(Flip\_Flop(C_0, I_0, T), R)$$

which is obtained by making the substitutions $C = C_0$, $I = I_0$, and $R_{T=0}$ and $R_{T=1}$ for $Q_{T=0}$ and $Q_{T=1}$ respectively in the right hand side of (7). Using

$$Q_{T=0} \equiv (C_1 \vee \neg I_0) \wedge (\neg C_1 \vee \neg I_1)$$

and

$$Q_{T=1} \equiv (C_1 \vee I_0) \wedge (\neg C_1 \vee I_1).$$

and substituting them in (7) gives:

$$\begin{aligned} R \equiv &\neg C_1 \vee \neg I_1 \vee [(C_1 \vee I_0) \wedge (\neg C_1 \vee I_1)] \\ &\wedge I_1 \vee [(C_1 \vee \neg I_0) \wedge (\neg C_1 \vee \neg I_1)] \\ &\wedge C_1 \vee [T \vee (C_1 \vee \neg I_0) \wedge (\neg C_1 \vee \neg I_1)] \\ &\wedge [\neg T \vee (C_1 \vee I_0) \wedge (\neg C_1 \vee I_1)]. \end{aligned}$$

$$\equiv (C_1 \vee T \vee \neg I_0) \wedge (\neg T \vee C_1 \vee I_0)$$

whence

$$R_{T=0} \equiv C_1 \vee \neg I_0$$

and

$$R_{T=1} \equiv C_1 \vee I_0$$

Finally (7) gives:

$$\begin{aligned} WP(S, Q) &\equiv \neg C_0 \vee [(\neg I_0 \vee R_{T=1}) \wedge (I_0 \vee R_{T=0})] \\ &\quad \wedge C_0 \vee [(T \vee R_{T=0}) \wedge (\neg T \vee R_{T=1})] \\ &\equiv \neg C_0 \vee [(\neg I_0 \vee C_1 \vee I_0) \wedge (I_0 \vee C_1 \vee \neg I_0)] \\ &\quad \wedge C_0 \vee [(T \vee C_1 \vee \neg I_0) (\neg T \vee C_1 \vee I_0) \\ &\equiv C_0 \vee C_1 \vee (T \wedge I_0) \vee (\neg T \wedge \neg I_0) \\ &\equiv C_0 \vee C_1 \vee (T = I_0). \end{aligned} \tag{8}$$

The precondition P (2) is that C_0 should have value 1 : $P \equiv C_0$. Since $C_0 \Rightarrow$ (8), then:

$$\{C_0\}\ \text{Flip\_Flop}(C_0, I_0, T);\ \text{Flip\_Flop}\ (C_1, I_1, T)\ \{Q\}$$

as required.

This is a nice example of a weakest precondition (8) which is genuinely weaker than the required precondition P. Students should work out for themselves why this is the case! The above discussion pays no attention to issues in parameter passing; if students encounter this more advanced topic, the flip-flop example might again provide an exercise.

Reference

[1] Backhouse, R.C. *Program Construction and Verification*, Prentice Hall International (UK) Ltd, 1986