FILE

[illegible]

IBM LABORATORY VIENNA, Austria

A PROOF OF THE CORRECTNESS OF SOME OPTIMIZING TECHNIQUES

C.B.JONES

ABSTRACT:
The application of certain well known optimizing techniques, to a very simple language, is proved to be correct. The proofs are based on properties of the language which, following Hoare, are shown as Axiom schema of statements.

LN 25.3.051
24 June 1969

1. INTRODUCTION

This note looks at certain optimizing techniques and attempts to show that their application to a program yields a modified program which, given any set of input values, will create the same output values as the original. The following techniques are commonly used (e.g. Ref. 4):

1. Elimination of common sub-expressions
2. Interchange of statements
3. Removal of statements from loops
4. Removal of statements from conditionals
5. Rearrangement of conditionals
6. Substitution of equivalent expressions
7. Reduction of slow operations such as multiply to faster ones within for loops
8. Commoning of "subsumed variables" (e.g. registers)

The first 6 of these are easily specified in the source language and are considered in section 3.

2. THE LANGUAGE

Programs of the language conform to the following abstract syntax:

```
is-program = is-st-list
is-st = is-null-st v is-as-st v is-cond-st v is-it-st
is-null-st = is-NULL
is-as-st = (<s-l-pt:is-var>,<s-r-pt:is-expr>)
is-cond-st = (<s-cond:is-b-expr>,<s-then-st:is-st-list>,
              <s-else-st:is-st-list>)
is-it-st = (<s-while-cl:is-b-expr>,<s-body:is-st-list>)
```

The important omissions are any constructs which could cause side effects and goto statements. It should be noted that all of the proofs given below rely on these assumptions.

All of the proofs are based on "properties of the statements of the language". These properties are expressed as axiom schema which are taken from Refs./1/ and /2/. They are all written in the notation of Ref. /2/.

A1 is-null-st

$P\{\emptyset\}P$

A2 is-as-st

$P(f,V)\{x:=f\}\ P(x,V)$

A3 is-cond-st

$((P \wedge b\{A\}R) \wedge (P \wedge \neg b\{B\}S)) \supset P\ \{\underline{\text{if}}\ b\ \underline{\text{then}}\ A\ \underline{\text{else}}\ B\}\ R \vee S$

A4 is-it-st

$(R \wedge b\{A\}R) \supset R\ \{\underline{\text{while}}\ b\ \underline{\text{do}}\ A\}\ \ R \wedge \neg b$

A5 is-st-list

$(P\{A\}Q \wedge Q\{B\}R) \supset P\{A;B\}R$

R1 Rule of consequence

if $P \supset Q$ and $Q\{A\}R$ can be derived, then $P\{A\}R$.

Note on the axiom of assignment:

The axiom A2 is derived from Igarashi (Ref./3/), and is rather more convenient for the construction of proofs than the form given in Floyd (Ref./1/):

A2' $P(x,V) \{x:=f(x,F)\} \exists x_o.(P(x_o,V) \wedge x = f(x_o,F))$

We show the equivalence of the two forms as follows:

Theorem: $A2 \vdash P \{x:=f\} Q$ iff $A2' \vdash P \{x:=f\} Q$

i) if $A2 \vdash P \{x:=f\} Q$ then $A2' \vdash P \{x:=f\} Q$

$P(f(x,F),V) \{x:=f(x,F)\} P(x,V)$ A2

now taking $P(x,V)$ as $P'(f(x,F),V)$ we can write.

$P'(f(x,F),V) \{x:=f(x,F)\} \exists x_o.(P'(f(x_o,F),V) \wedge x = f(x_o,F))$ A2'

$\therefore P'(f(x,F),V) \{x:=f(x,F)\} P'(x,V)$

Thus if $A2 \vdash P \{x:=f\} Q$ then $A2' \vdash P \{x:=f\} Q$

ii) if $A2' \vdash P \{x:=f\} Q$ then $A2 \vdash P \{x:=f\} Q$

$P(x,V) \{x:=f(x,F)\} \exists x_o.(P(x_o,V) \wedge x = f(x_o,F))$ A2'

now taking $P(x,V)$ as $\exists x_o.P'(x_o,V) \wedge x = f(x_o,V)$ we can write

$\exists x_o.(P'(x_o,V) \wedge f(x,F) = f(x_o,F)) \{x:=f(x,F)\}$

$\exists x_o.(P'(x_o,V) \wedge x = f(x_o,F))$ A2

now $P'(x,V) \supset \exists x_o.P'(x_o,V)$

and $T \supset \exists x_o.f(x,F) = f(x_o,F)$

$\therefore P'(x,V) \{x := f(x,F)\} \exists x_o.(P'(x_o,V) \wedge x = f(x_o,F))$ R1

Thus if $A2' \vdash P \{x:= f\}Q$ then $A2 \vdash P\{x:= f\}Q$

Combining these two results we get $A2 \vdash P\{x := f\}Q$ iff $A2' \vdash P\{x :=f\}Q$

QED

3. PROOFS

Each optimizing technique to be discussed will be exhibited by a pair of schema, into which substitution of elements satisfying the preconditions is alleged to yield a pair of equivalent sequences of statements. By "equivalent" we mean that after execution of one of the sequences of statements, on any environment, it is not possible to determine which sequence was executed. We write this relation $L \cong R$.

Our proofs of these conjectures are made by showing that if any ordered pair of predicates are respectively true before and after execution of one of the sequences they will also be true before and after the other. The free variables of the predicates being chosen from the variables of the program. Or, using "$\vdash$" to denote "can be derived from the axioms":

$$\forall P,Q.\ (\vdash P\{L\}Q \ \text{ iff } \vdash P\{R\}Q)$$

We shall use the following notational conventions:

P,Q,R,S	Predicates
A,B,C	Arbitrary sequences of statements
A(f)	A(x) with f substituted for every free occurence of x
L[A]	The set of all variables occuring on the left of assignment statements in A
R[A]	The set of all variables occuring in expressions of A
V[A]	$L[A] \cup R[A]$
$\mathcal{V}$	The set of all free variables of both statement sequences
$\mathcal{A}$(S)	The function which is considered to show the transformation A(S) causes to L[A(S)] thus $A(S) \cong L[A(S)] := \mathcal{A}(S)$

In deriving sequences of sentences from the axioms the predicate on the right of } is sometimes omitted, the implied predicate is that on the left of { in the preceding line.

1: Elimination of common sub-expressions

$$A(f) \cong t := f;\ A(t) \quad \text{where} \quad (t \cup V[f]) \cap L[A(t)] = \emptyset$$

This can in fact be made less restrictive by allowing the last statement, in which the substitution is performed, to change a free variable of f.

$$A(f);\ x := g(f) \cong t := f;\ A(t);\ x := g(t)$$
$$\text{where } (t \cup V[f]) \cap L[A(t)] = \emptyset \text{ and } x \in V[f]$$

Let

$V[t] = t$

$V[F] - x = F$

$L[A(t)] = L$

$\mathcal{V} - (t \cup F \cup L) = V$

Theorem: $A(f(x,F),x,F,L,V);\ x := g(f(x,F),x,F,L,V)$
$\cong t := f(x,F);\ A(t,x,F,L,V);\ x := g(t,x,F,L,V)$

Proof:

L) $Q(g(f(x,F),x,F,L,V),F,L,V)\ \{x := g(f(x,F),x,F,L,V)\}\ Q(x,F,L,V)$ A2

$Q(g(f(x,F),x,F,\mathcal{A}(f(x,F),x,F,L,V),V),F,\mathcal{A}(f(x,F),x,F,L,V),V)$
$\{A(f(x,F),x,F,L,V)\}$ A2

R) $Q(g(t,x,F,L,V),F,L,V) \wedge t = f(x,F)\ \{x := g(t,x,F,L,V)\}\ Q(x,F,L,V) \wedge$
$\exists x_o.(t = f(x_o,F) \wedge x = g(t,x_o,F,L,V))$ A2,R1

$Q(g(t,x,F,\mathcal{A}(t,x,F,L,V),V)F,\mathcal{A}(t,x,F,L,V),V) \wedge t = f(x,F)$
$\{A(t,x,F,L,V)\}$ A2

$Q(g(f(x,F),x,F,\mathcal{A}(f(x,F),x,F,L,V),V),F,\mathcal{A}(f(x,F),x,F,L,V),V)$
$\{t := f(x,F)\}$ A2

Thus ignoring the information about the temporary,

$$\vdash P\ \{L\}\ Q \qquad \text{iff} \qquad \vdash P\ \{R\}\ Q$$ A5

Notice that if we allow f to be moved out of the range of a conditional statement or a while statement it may be evaluated with values of the free variables that would not be used in A because of the conditional. Thus f must be a total function.

It is, in theory, possible to avoid this problem but, since the set of predicates which govern execution of a statement may contain variables whose values change, it would probably make the necessary tests too expensive.

2: Move calculations to beginning of loops

$A;B \cong B;A$ where $L[A] \cap L[B] = L[A] \cap R[B] = L[B] \cap R[A] = \emptyset$

Let $L[A] = LA$

$L[B] = LB$

$(R[A] - LA) \cup (R[B] - LB) = R$

<u>Theorem:</u> $A(LA,R);\ B(LB,R) \cong B(LB,R);\ A(LA,R)$

<u>Proof:</u>

L) $Q(LA,\mathcal{B}(LB,R),R)\quad \{B(LB,R)\}\ Q(LA,LB,R)$ A2

$Q(\mathcal{A}(LA,R),\mathcal{B}(LB,R),R)\quad \{A(LA,R)\}$ A2

R) $Q(\mathcal{A}(LA,R),B,R)\quad \{A(LA,R)\}\ Q(LA,LB,R)$ A2

$Q(\mathcal{A}(LA,R),\mathcal{B}(LB,R),R)\quad \{B(LB,R)\}$ A2

Thus $\vdash P\ \{L\}\ Q$ iff $\vdash P\ \{R\}\ Q$ A5

3: Removal of statements from loops

B; <u>while</u> p <u>do</u> A $\cong$ <u>while</u> p <u>do</u> B; A

<u>where</u> $L[A] \cap L[B] = L[A] \cap R[B] = L[B] \cap V[p] = \emptyset$

Let $L[A] = LA$

$L[B] = LB$

$(R[A] \cup R[B]) - (LA \cup LB) = R$

$V[p] - LA - R = VP$

<u>Theorem:</u> B(LB,R); <u>while</u> p(LA,R,VP) <u>do</u> A(LA,LB,R)

$\cong$ <u>while</u> p(LA,R,VP) <u>do</u> B(LB,R); A (LA,LB,R)

<u>Proof:</u>

L) We assume the strongest invariant for the loop is such that

H1 $\vdash Q(LA,LB,R,VP) \wedge LB = \mathcal{B}(LB,R) \wedge p(LA,R,VP)$ {A(LA,LB,R)}

$Q(LA,LB,R,VP) \wedge LB = \mathcal{B}(LB,R)$

$\therefore Q(LA,LB,R,VP) \wedge LB = \mathcal{B}(LB,R)$ { <u>while</u> p(LA,R,VP) <u>do</u> A(LA,LB,R)} A4

$Q(LA,LB,R,VP) \wedge LB = \mathcal{B}(LB,R) \wedge \neg p(LA,R,VP)$

$Q(LA,\mathcal{B}(LB,R),R,VP)$ {B(LB,R)} A2

R) From H1

$Q(LA,\mathcal{B}(LB,R),R,VP) \wedge p(LA,R,VP)$ {B(LB,R); A(LA,LB,R)} A2,A5

$Q(LA,LB,R,VP) \wedge LB = \mathcal{B}(LB,R)$

$\therefore Q(LA,\mathcal{B}(LB,R),R,VP)$ { <u>while</u> p(LA,R,VP) <u>do</u> B(LB,R); A(LA,LB,R)} A4

$Q(LA,LB,R,VP) \wedge LB = \mathcal{B}(LB,R) \wedge \neg p(LA,R,VP)$

Thus we have shown

if $\vdash P \{L\} Q$ then $\vdash P \{R\} Q$ A5

Using the first line of R above as an invariant, we can show, via A2':

if $\vdash P \{R\} Q$ then $\vdash P \{L\} Q$

Therefore,

$\vdash P \{L\} Q$ iff $\vdash P \{R\} Q$

4: Removal of statements from arms of a conditional

a) $A;\ \underline{\text{if}}\ p\ \underline{\text{then}}\ B\ \underline{\text{else}}\ C \quad \cong\ \underline{\text{if}}\ p\ \underline{\text{then}}\ A;\ B\ \underline{\text{else}}\ A;\ C$

where $L[A] \cap V[p] = \emptyset$

Let $L[A] = LA$

$\mathcal{V} - LA = V$

<u>Theorem</u>: $A(LA,V);\ \underline{\text{if}}\ p(V)\ \underline{\text{then}}\ B(LA,V)\ \underline{\text{else}}\ C(LA,V)$

$\cong\ \underline{\text{if}}\ p(V)\ \underline{\text{then}}\ A(LA,V);\ B(LA,V)\ \underline{\text{else}}$

$A(LA,V);\ C(LA,V)$

<u>Proof:</u>

L) We first assume

H1 $P(LA,V) \wedge p(V)\ \{B(LA,V)\}\ R$

H2 $P(LA,V) \wedge \neg p(V)\ \{C(LA,V)\}\ S$

$\therefore P(LA,V)\ \{\underline{\text{if}}\ p(V)\ \underline{\text{then}}\ B(LA,V)\ \text{else}\ C(LA,V)\}\ R \vee S$ A3

$P(\mathcal{A}(LA,V),V)\quad \{A(LA,V)\}\ P(LA,V)$ A2

R) From H1 $P(\mathcal{A}(LA,V),V) \wedge p(V) \quad \{A(LA,V);\ B(LA,V)\}\ R$ A5,A2

From H2 $P(\mathcal{A}(LA,V),V) \wedge \neg p(V) \quad \{A(LA,V);\ C(LA,V)\}\ S$ A5,A2

$\therefore P(\mathcal{A}(LA,V),V)\ \{\underline{\text{if}}\ p(V)\ \underline{\text{then}}\ A(LA,V);\ B(LA,V)\ \underline{\text{else}}\ A(LA,V);$

$C(LA,V)\}\ R \vee S$ A3

Thus $\vdash P\ \{L\}\ Q$ iff $\vdash P\ \{R\}\ Q$ A5

b) In the following case it is possible to remove the statement even if it affects p.

Let $\mathcal{V} - x = V$

<u>Theorem</u>:

$x := f(x,V);\ \underline{\text{if}}\ p(x)\ \underline{\text{then}}\ A(x,V)\ \underline{\text{else}}\ B(x,V)$

$\cong\ \underline{\text{if}}\ p(f(x,V))\ \underline{\text{then}}\ x := f(x,V);\ A(x,V)\ \underline{\text{else}}$

$x := f(x,V);\ B(x,V)$

Proof: We first assume

L) H1 $P(x,V) \wedge p(x) \{A(x,V)\} R$
H2 $P(x,V) \wedge \neg p(x) \{B(x,V)\} S$

$\therefore P(x,V) \{$ if $p(x)$ then $A(x,V)$ else $B(x,V)\}$ $R \vee S$ A3
$P(f(x,V),V) \{x := f(x,V)\}$ $P(x,V)$ A2

R) From H1 $P(f(x,V),V) \wedge p(f(x,V)) \{x := f(x,V); A(x,V)\} R$ A5,A2
From H2 $P(f(x,V),V) \wedge \neg p(f(x,V)) \{x:=f(x,V); B(x,V)\} S$ A5,A2

$\therefore P(f(x,V),V) \{$if $p(f(x,V))$ then $x := f(x,V); A(x,V)$ else
$x := f(x,V); B(x,V)\} R \vee S$ A3
Thus $\vdash P \{L\} Q$ iff $\vdash P \{R\} Q$ A5

5: If statements

The object of performing these transformations is to avoid evaluating boolean expressions completely where possible. The transformations are shown in terms of copying the text although a real compiler would use goto statements.

a) if $p \wedge q$ then A else B $\cong$ if $\neg p$ then B else (if q then A else B)

Proof:
L) We first assume:
H1 $P \wedge (p \wedge q) \{A\} R$
H2 $P \wedge \neg(p \wedge q) \{B\} S$

$\therefore P \{$ if $p \wedge q$ then A else B$\} R \vee S$ A3

R) $P \wedge p \wedge q \{A\} R$ H1
$P \wedge p \wedge \neg q \{B\} S$ H2
$\therefore P \wedge p \{$ if q then A else B$\} R \vee S$ A3
$P \wedge \neg p \{B\} S$ H1
$\therefore P \{$ if $\neg p$ then B else (if q then A else B)$\} R \vee S$ A3

Thus $\vdash P \{L\} Q$ iff $\vdash P \{R\} Q$

b) <u>if</u> $p \vee q$ <u>then</u> A <u>else</u> B $\cong$ <u>if</u> p <u>then</u> A <u>else</u> (<u>if</u> q <u>then</u> A <u>else</u> B)

<u>Proof:</u>

L) We first assume:

H1 $P \wedge (p \vee q)$ {A} R

H2 $P \wedge \neg(p \vee q)$ {B} S

$\therefore$ P {<u>if</u> $p \vee q$ <u>then</u> A <u>else</u> B} $R \vee S$ A3

R) $P \wedge \neg p \wedge \neg q$ {B} S H2

$P \wedge \neg p \wedge q$ {A} R H1

$\therefore P \wedge \neg p$ {<u>if</u> q <u>then</u> A <u>else</u> B} $R \vee S$ A3

$P \wedge p$ {A} R H1

$\therefore$ P { <u>if</u> p <u>then</u> A <u>else</u> (<u>if</u> q <u>then</u> A <u>else</u> B)} $R \vee S$ A3

Thus $\vdash$ P {L} Q iff $\vdash$ P {R} Q

6: Substitution of expressions

Some examples of when a compiler might use such substitutions are

i) special cases of exponentiation
ii) division by constants changed to multiplication
iii) use of reordered predicates

a) If f = g, then x := L(f) $\cong$ x := L(g)

<u>Proof</u>:

L) P(L(f),V) {x := L(f)} P(x,V)

R) P(L(g),V) {x := L(g)} P(x,V)

but P(L(g),V) $\equiv$ P(L(f),V)

thus $\vdash$ P {x:= L(f)} Q iff $\vdash$ P {x:= L(g)} Q R1

b) If $p \equiv q$ then $\underline{\text{if}}\ p\ \underline{\text{then}}\ A\ \underline{\text{else}}\ B \cong \underline{\text{if}}\ q\ \underline{\text{then}}\ A\ \underline{\text{else}}\ B$

<u>Proof</u>:

L) We assume

H1 $\quad P \wedge p\ \{A\}\ R$

H2 $\quad P \wedge \neg p\ \{B\}\ S$

$\therefore P\ \{\underline{\text{if}}\ p\ \underline{\text{then}}\ A\ \underline{\text{else}}\ B\}\ R \vee S$ A3

R) now $P \wedge q \equiv P \wedge p$ and $P \wedge \neg q = P \wedge \neg p$

$\therefore P \wedge q\ \{A\}\ R$ H1,R1

and $\quad P \wedge \neg q\ \{B\}\ S$ H2,R1

$\therefore P\ \{\underline{\text{if}}\ p\ \underline{\text{then}}\ A\ \underline{\text{else}}\ B\}\ R \vee S$ A3

Thus $\vdash P\ \{L\}\ Q \quad$ iff $\quad \vdash P\ \{R\}\ Q$

Clearly a similar proof permits substitution of equivalent predicates in a while clause.

c) If $p \supset f = g$ then $\underline{\text{if}}\ p\ \underline{\text{then}}\ x := f;\ A\ \underline{\text{else}}\ B \cong \underline{\text{if}}\ p\ \underline{\text{then}}\ x := g;\ A\ \underline{\text{else}}\ B$

<u>Proof</u>: Follows from:

L) $P(f(x,V),V) \wedge p(f(x,V),V)\ \{x := f(x,V)\}\ P(x,V) \wedge p(x,V)$ A2

R) $P(g(x,V),V) \wedge p(g(x,V),V)\ \{x := g(x,V)\}\ P(x,V) \wedge p(x,V)$ A2

$p(f(x,V),V) \equiv P(g(x,V),V)$

$p(g(x,V),V) \supset P(g(x,V),V)$

$\therefore P(f(x,V),V) \wedge p(f(x,V),V) \equiv P(g(x,V),V) \wedge p(g(x,V),V)$

Thus $\vdash P\ \{L\}\ Q \quad$ iff $\quad \vdash P\ \{R\}\ Q$

Clearly b can be extended to the case where the equivalence is implied by a predicate used in a conditional. Both this and c could then be further extended to cover a nest of such preconditions or a sequence of intervening statements which do not change any of the free variables of the predicate.

4. SUMMARY

The approach used in section 3. appears to yield straightforward proofs for a number of optimizing techniques. The problem of extending the work is more in stating the optimization in the source language than in costructing the proof.

It is worth noting that a program with its set of predicates would be a very good guide to an optimizing compiler in showing exactly what relationsips must be preserved.

REFERENCES

/1/ R.W.FLOYD: Assigning Meaning to Programs.- 20 May 1966.

/2/ C.A.R.HOARE: The Axiomatic Basis of Computer Programming.- To be published.

/3/ S.IGARASHI: Equivalence-Theoretical Treatment of Verification Conditions.-

/4/ E.S.LOWRY and C.W.MEDLOCK: Object Code Optimization.- Comm.ACM 12 (1969) No. 1.