Tony Maldonado
September 9, 2020
CS 471 – Program 2

Programming #2 – Short Circuit Evaluation

Problem Description:

- Some programming languages are implemented in such a manner that in the AND boolean construct there is a short circuit evaluation. Essentially, short circuit evaluation is when in an AND boolean expression the language evaluates only the first part of the expression and if it knows the result of the expression, skips evaluating the second part of the expression. This happens when if you're evaluating A && B, A is false. A being false makes the whole expression be false, so the language skips over evaluating the second part since because regardless of its boolean value, the full expression will be false.
- It turns out, all four of the languages we tested had some sort of short circuit evaluation:

Results:

Language	Short Circuit?	Notes/Comments
ADA	Yes	Has short circuit only with the 'and then' operator
Perl	Yes	
PHP	Yes	
Shell	Yes	I used BASH (Bourne) instead of C-Shell

Perl

```
#!/usr/bin/perl

# Name: Tony Maldonado
# Date: September 09, 2020
#
# Input: None
# Output: Whether the second condition is evaluated or not after
#         the first condition in an AND conditional statement
#         evaluates as False
#
# Preconditions: None
# Postconditions: None
```

```
# Function that will be used for testing. If visited, then
# it will print the message, if not visited, then it won't
# print the message and we will know there was a short circuit
sub evaluate {
  printf "Function visited\n";
  return 1;
}

$a = 1;

printf "\n";
printf "First testing with the variable as first condition\n";

# Testing with variable as first condition
# Test for both conditions as True
printf "\n";
printf "Conditions: T and T\n";
if ($a == 1 && evaluate() ) {
  printf "True\n";
} else {
    printf "False\n";
}

# Now with False && True
# The following test shows that Perl DOES short circuit
# because $a is NOT 0 therefore the first of the if
# statement is FALSE and since in the output we don't see
# the print statement from evaluate(), we know it doesn't
# check after evaluating the first as a false.
printf "\n";
printf "Conditions: F and T\n";
if ($a == 0 && evaluate() ) {
  printf "True\n";
} else {
    printf "False\n";
}

printf "\n";

printf "Now testing with the function as first condition\n";
# First True && True
printf "\n";
printf "Conditions: T and T\n";
if (evaluate() && $a == 1) {
```

```
  printf "True\n";
} else {
    printf "False\n";
}

# Then False && True
printf "\n";
printf "Conditions: F and T\n";
if (!evaluate() && $a == 1) {
  printf "True\n";
} else {
    printf "False\n";
}

printf "\n";
```

Perl – Output

```
ssh amaldona@newton.cs.nmsu.edu — -ssh amaldona@newton
newton cs471/program2> perl program2.pl

First testing with the variable as first condition

Conditions: T and T
Function Evaluated
True

Conditions: F and T
False

Now testing with the function as first condition

Conditions: T and T
Function Evaluated
True

Conditions: F and T
Function Evaluated
False

newton cs471/program2>
```

Tony Maldonado
September 9, 2020
CS 471 – Program 2

Ada

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;

procedure program2 is
  A : Integer;

  -- Function that will be used for testing. If visited, then
  -- it will print the message, if not visited, then it won't
  -- print the message and we will know there was a short circuit

  function Evaluation return Integer is
  begin
    Put_Line("Function visited");
    return 1;
  end Evaluation;

begin
  A := 1;

  -- We begin by testing the AND operator
  Put_Line("First we test the AND operator:");

  Put_Line("");
  Put_Line("First testing with the variable as first condition");
  Put_Line("Conditions: T and T");
  if A = 1 and Evaluation = 1 then
    Put_Line("True");
  else
    Put_Line("False");
  end if;

  -- This test should show us whether there is a short circuit
  -- If the message in the function doesn't get printed, then
  -- there is a short circuit in ada.

  Put_Line("");
  Put_Line("Conditions: F and T");
  if A = 0 and Evaluation = 1 then
    Put_Line("True");
  else
    Put_Line("False");
  end if;
```

```
Put_Line("");

Put_Line("");
Put_Line("Now testing with the function as first condition");
Put_Line("Conditions: T and T");
if Evaluation = 1 and A = 1 then
  Put_Line("True");
else
  Put_Line("False");
end if;

Put_Line("");
Put_Line("Conditions: F and T");
if Evaluation = 0 and A = 1 then
  Put_Line("True");
else
  Put_Line("False");
end if;

Put_Line("");

-- We now test the AND THEN operator
Put_Line("Now we test the AND THEN operator:");

Put_Line("");
Put_Line("First testing with the variable as first condition");
Put_Line("Conditions: T and then T");
if A = 1 and then Evaluation = 1 then
  Put_Line("True");
else
  Put_Line("False");
end if;

-- The following test should show us whether there is a short
-- circuit. If the message in the function doesn't get printed,
-- then there is a short circuit in ada.

Put_Line("");
Put_Line("Conditions: F and then T");
if A = 0 and then Evaluation = 1 then
  Put_Line("True");
else
  Put_Line("False");
end if;
```

```
  Put_Line("");

  Put_Line("");
  Put_Line("Now testing with the function as first condition");
  Put_Line("Conditions: T and then T");
  if Evaluation = 1 and then A = 1 then
    Put_Line("True");
  else
    Put_Line("False");
  end if;

  Put_Line("");
  Put_Line("Conditions: F and then T");
  if Evaluation = 0 and A = 1 then
    Put_Line("True");
  else
    Put_Line("False");
  end if;

end program2;
```

Tony Maldonado
September 9, 2020
CS 471 – Program 2

Ada – Output

```
ssh amaldona@newton.cs.nmsu.edu — -ssh amaldona@newton.cs.nmsu.edu — 107×48
newton cs471/program2> clear

newton cs471/program2> gcc -c program2.adb
newton cs471/program2> gnatmake --GNATBIND=gnatbind --GNATLINK=gnatlink program2
gnatbind -x program2.ali
gnatlink program2.ali
newton cs471/program2> ./program2
First we test the AND operator:

First testing with the variable as first condition
Conditions: T and T
Function evaluated
True

Conditions: F and T
Function evaluated
False


Now testing with the function as first condition
Conditions: T and T
Function evaluated
True

Conditions: F and T
Function evaluated
False

Now we test the AND THEN operator:

First testing with the variable as first condition
Conditions: T and then T
Function evaluated
True

Conditions: F and then T
False


Now testing with the function as first condition
Conditions: T and then T
Function evaluated
True

Conditions: F and then T
Function evaluated
False
newton cs471/program2>
```

Tony Maldonado
September 9, 2020
CS 471 – Program 2

PHP

```
<?php

// Function that will be used to evaluate second condition being looked at
function evaluate() {
  echo 'Function visited.'.PHP_EOL;
  return true;
}

echo ''.PHP_EOL;
echo 'First testing with the variable as first condition'.PHP_EOL;

$A = 1;

// Testing with variable as first condition: T && T
echo 'Conditions: T and T'.PHP_EOL;
if ( $A == 1 && evaluate() ) {
  echo 'True'.PHP_EOL;
} else {
    echo 'False'.PHP_EOL;
}

// F && T
// This test should show us whether there is a short circuit
// If the message in the function doesn't get printed, then
// there is a short circuit in php.
echo ''.PHP_EOL;
echo 'Conditions: F and T'.PHP_EOL;
if ( $A == 0 && evaluate() ) {
  echo 'True'.PHP_EOL;
} else {
    echo 'False'.PHP_EOL;
}

echo ''.PHP_EOL;
echo 'Now testing with the function as first condition'.PHP_EOL;
echo 'Conditions: T and T'.PHP_EOL;
if ( evaluate() && $A == 1 ) {
  echo 'True'.PHP_EOL;
} else {
    echo 'False'.PHP_EOL;
}

echo ''.PHP_EOL;
```

```
echo 'Conditions: F and T'.PHP_EOL;
if ( !evaluate() && $A == 1 ) {
  echo 'True'.PHP_EOL;
} else {
    echo 'False'.PHP_EOL;
}

?>
```

PHP – Output

```
ssh amaldona@newton.cs.nmsu.edu — -ssh amaldona@newto
newton cs471/program2> php program2.php

First testing with the variable as first condition
Conditions: T and T
Function visited.
True

Conditions: F and T
False

Now testing with the function as first condition
Conditions: T and T
Function visited.
True

Conditions: F and T
Function visited.
False
newton cs471/program2>
```

Tony Maldonado
September 9, 2020
CS 471 – Program 2

Bourne Shell:

```
#!/bin/sh

a=1

# Rather than using a function as the second condition,
# I will be using a simple 'echo' statement to test whether
# it gets evaluated after the first condition is False

# First test with variable as first condition
# True && True
echo ""
echo "First testing with the variable as first condition"
echo "Conditions: T and T"
if [[ $a == 1 ]] && echo "Visited"
then
  echo "True"
else
  echo "False"
fi

# False && True
# This test should show us whether there is a short circuit
# If the echo statement doesn't get printed, then
# there is a short circuit in shell.
echo ""
echo "Conditions: F and T"
if [[ $a == 0 ]] && echo "Visited"
then
  echo "True"
else
  echo "False"
fi

# Then test with echo first
# True && True
echo ""
echo "Now testing with the echo as first condition"
echo "Conditions: T and T"
if echo "Visited" && [[ $a == 1 ]]
then
  echo "True"
```

```
else
  echo "False"
fi

# False && True
echo ""
echo "Conditions: F and T"
if echo "Visited" && [[ $a == 0 ]]
then
  echo "True"
else
  echo "False"
fi
```

Bourne Shell - Output

```
ssh amaldona@newton.cs.nmsu.edu — -ssh amaldona@newton.cs.nmsu.edu — 68×19
newton cs471/program2> ./program2.sh

First testing with the variable as first condition
Conditions: T and T
Visited
True

Conditions: F and T
False

Now testing with the echo as first condition
Conditions: T and T
Visited
True

Conditions: F and T
Visited
False
newton cs471/program2>
```