Warren's Abstract Machine

Chapters 2 and 3

Exercises

1 Recognize the Representation

Exercise 1.1 : Terms, Functors, and Arity

For each expression, identify the outer functor and arity, and list the variables.

1. $f(X, a)$
2. $g(h(X), Y, b)$
3. $p(Z, h(Z, W), f(W))$

Exercise 1.2 : Fill in the Heap

Complete one valid representation of:

$$f(X, a).$$

```
Address    Contents
0          <STR,1>
1          ----------
2          ----------     <- X
3          <STR,4>
4          ----------
```

Exercise 1.3 : Shared Variable

Construct a valid heap representation for:

$$g(X, X).$$

Exercise 1.4 : Dereference

Before completing this exercise, recall the `deref` operation from the WAM book.

The `deref` Operation

```
function deref(a : address) : address;
begin
    <tag, value> <- STORE[a];
```

```
    if (tag = REF) and (value != a)
    then return deref(value)
    else return a
end deref;
```

Suppose the heap contains:

```
HEAP[4]  = <REF,9>
HEAP[9]  = <STR,20>
HEAP[12] = <REF,15>
HEAP[15] = <REF,15>
HEAP[18] = <STR,25>

HEAP[20] = f/1
HEAP[25] = g/2
```

Compute:

$$\text{deref}(4), \qquad \text{deref}(12), \qquad \text{deref}(15), \qquad \text{deref}(18).$$

For each call, show the addresses visited by `deref`.

Exercise 1.5 : Pick the Instruction

Fill in the instruction name for $\mathcal{M}_0$.

Situation	`Instruction`
Query: register associated with a structure functor	_____
Query: first-seen register argument	_____
Query: already-seen register argument	_____
Program: register associated with a structure functor	_____
Program: first-seen register argument	_____
Program: already-seen register argument	_____

2 Compile and Trace $\mathcal{M}_0$

Exercise 2.1 : Guided $\mathcal{M}_0$ Query Compilation

Compile:

```
?- p(Z, h(Z,W), f(W)).
```

Step 1: Register allocation
Fill in:

$$X_1 = p(_____, _____, _____),$$
$$X_2 = _____,$$
$$X_3 = h(_____, _____),$$
$$X_4 = f(_____),$$
$$X_5 = _____.$$

Step 2: Put the flattened query in bottom-up order

_____ → _____ → _____

Step 3: Fill in the instructions

```
put_structure h/2, ______
set_variable ______
set_variable ______

put_structure f/1, ______
set_value ______

put_structure p/3, ______
set_value ______
set_value ______
set_value ______
```

Step 4: Build the heap
Starting with:

$$H = 0,$$

give the final heap and mark which addresses represent Z and W.
Step 5: Check
Does the final heap represent exactly:

$$p(Z, h(Z, W), f(W))?$$

Explain how the two occurrences of Z share one variable, and likewise for W.

Exercise 2.2 : Guided $\mathcal{M}_0$ Program Compilation

Compile:

```
p(f(X), h(Y,f(a)), Y).
```

Step 1: Register allocation
Complete a valid allocation:

$$X_1 = p(X_2, X_3, X_4),$$
$$X_2 = _____,$$
$$X_3 = _____,$$
$$X_4 = _____,$$
$$X_5 = _____,$$
$$X_6 = _____,$$
$$X_7 = _____.$$

Step 2: Order the program top-down

Step 3: Fill the instruction types

```
__________ p/3, X1
__________ X2
__________ X3
__________ X4

__________ f/1, X2
__________ X5

__________ h/2, X3
__________ X4
__________ X6

__________ f/1, X6
__________ X7

__________ a/0, X7
```

Step 4: Explain
Which instructions can enter read or write mode?
What determines the mode?

Exercise 2.3 : Trace General Unification

Adapted from Exercise 2.2 of the WAM book.
Give heap representations for:

$$f(X, g(X, a))$$

and:

$$f(b, Y).$$

Let a_1 and a_2 be their respective heap addresses.
Let a_X and a_Y denote the heap addresses corresponding to variables X and Y.
Trace:

$$\text{unify}(a_1, a_2).$$

Your trace must show:

1. the pairs processed from the PDL;
2. the dereferenced addresses;
3. each call to `bind`;
4. every heap cell that changes;
5. the final dereferenced bindings for X and Y.

Exercise 2.4 : Does It Unify?

For each pair, state whether unification succeeds.
If it succeeds, give an MGU.

1. $f(X, a)$ and $f(b, Y)$

2. $f(X, X)$ and $f(a, b)$
3. $g(X, h(Y))$ and $g(f(a), h(b))$
4. $f(X)$ and $g(X)$

3 Complete $\mathcal{M}_0$ Executions

Exercise 3.1 : Query Followed by Program

Adapted from Exercise 2.3 of the WAM book.

$\mathcal{L}_0$ Query

```
?- p(Z, h(Z,W), f(W)).
```

$\mathcal{L}_0$ Program

```
p(f(X), h(Y,f(a)), Y).
```

Compile and execute the query, followed immediately by the program.
Do not reset H between the query and program executions.
Give the final dereferenced bindings for:

$$W, \qquad X, \qquad Y, \qquad Z.$$

Apply the bindings to both the query term and the program term.
Verify that the resulting terms are identical.

Exercise 3.2 : Reverse Query and Program

Adapted from Exercises 2.4 and 2.5 of the WAM book.

$\mathcal{L}_0$ Query

```
?- p(f(X), h(Y,f(a)), Y).
```

$\mathcal{L}_0$ Program

```
p(Z, h(Z,W), f(W)).
```

Compile the query and the program into $\mathcal{M}_0$ instructions.
Execute the program immediately after the query without resetting H.
Give the final MGU.
Does the reversed query/program setup produce the same MGU as Exercise 3.1?

4 Transition to $\mathcal{M}_1$

From $\mathcal{L}_0/\mathcal{M}_0$ to $\mathcal{L}_1/\mathcal{M}_1$

Key Changes

	$\mathcal{L}_0/\mathcal{M}_0$	$\mathcal{L}_1/\mathcal{M}_1$
Program	One program term.	A set of first-order atoms, with at most one fact per predicate name.
Predicate symbol	Part of the term represented on the heap.	Eliminated from the heap; the predicate name identifies an instruction sequence in `CODE`.
Term roots	One term root.	One term root for each predicate argument, held in $A_1, \ldots, A_n$.
Predicate call	No separate call instruction.	Use `call p/n`.
End of fact	No separate instruction.	Use `proceed`.
Variable arguments	Not applicable: there are no argument registers.	Use `put_variable`, `put_value`, `get_variable`, `get_value`.

Exercise 4.1 : Argument-Register Allocation

Complete the argument-register allocation for each atom.

(a)

$$p(Z, h(Z, W), f(W))$$

Complete:

$$A_1 = _____,$$
$$A_2 = h(_____, _____),$$
$$A_3 = f(_____),$$
$$X_4 = _____.$$

(b)

$$p(h(Z, W), Z, f(W))$$

Complete:

$$A_1 = h(_____, _____),$$
$$A_2 = _____,$$
$$A_3 = f(_____),$$
$$X_4 = _____,$$
$$X_5 = _____.$$

How does the position of the first occurrence of Z affect its register allocation?

Exercise 4.2 : Instructions for Variable Arguments

For each case, choose one of:

`put_variable`, `put_value`, `get_variable`, `get_value` .

1. Query `?- p(X,...)`. This is the first occurrence of X.
2. Query `?- p(X,X)`. Consider the second argument.
3. Fact `p(X,...)`.. This is the first occurrence of X.
4. Fact `p(X,X)`.. Consider the second argument.

5 Compile and Trace $\mathcal{M}_1$

Exercise 5.1 : Build the $\mathcal{M}_1$ Query

Adapted from Exercise 2.6 of the WAM book.

For:

```
?- p(Z, h(Z,W), f(W)).
```

execute:

```
put_variable  X4, A1
put_structure h/2, A2
set_value     X4
set_variable  X5
put_structure f/1, A3
set_value     X5
call          p/3
```

Starting at:

$$H = 0,$$

1. Show the heap after every heap-modifying instruction.
2. Give the final contents of:
$$A_1, A_2, A_3, X_4, X_5.$$
3. Compare the result with the $\mathcal{M}_0$ representation from Exercise M1.
4. Explain what happened to the outer predicate symbol `p/3`.

Exercise 5.2 : Execute the $\mathcal{M}_1$ Fact

Adapted from Exercise 2.7 of the WAM book.

Immediately after Exercise M5, execute:

```
p/3:
    get_structure f/1, A1
    unify_variable X4

    get_structure h/2, A2
    unify_variable X5
```

```
    unify_variable X6

    get_value X5, A3

    get_structure f/1, X6
    unify_variable X7

    get_structure a/0, X7

    proceed
```

Trace the relevant reads, writes, and bindings.
Give the final MGU and verify that the two original atoms become identical.

6 Design $\mathcal{M}_1$ Instructions Yourself

Exercise 6.1 : Reverse the $\mathcal{L}_1$ Query and Program

Adapted from Exercises 2.8 and 2.9 of the WAM book.

$\mathcal{L}_1$ Query

```
?- p(f(X), h(Y,f(a)), Y).
```

$\mathcal{L}_1$ Program

```
p(Z, h(Z,W), f(W)).
```

STEP 1: Query register allocation

$$
\begin{aligned}
A_1 &= f(_____),\\
A_2 &= h(_____, _____),\\
A_3 &= _____,\\
X_4 &= _____,\\
X_5 &= _____,\\
X_6 &= f(_____),\\
X_7 &= _____.
\end{aligned}
$$

STEP 2: Query instructions

Fill in the blanks.

```
put_structure f/1, A1
set_variable ______

put_structure a/0, ______

put_structure f/1, ______
set_value ______

put_structure h/2, A2
set_variable ______
set_value ______

put_value ______, A3

call p/3
```

STEP 3: Program instructions
Fill in the blanks.

```
p/3:
    get_variable ______, A1

    get_structure h/2, A2
    unify_value ______
    unify_variable ______

    get_structure f/1, A3
    unify_value ______

    proceed
```

STEP 4: Verification
Execute the code.
Does it produce the same MGU as Exercise M6?
Why should reversing the query and program terms not alter the MGU?

7 Concept Review

Read Mode or Write Mode?

Suppose:

```
get_structure f/2, X3
```

is executed.
For each state, say whether the instruction enters read mode, write mode, or fails.

1. deref(X_3) reaches an unbound $\langle \text{REF}, 12 \rangle$.
2. deref(X_3) reaches $\langle \text{STR}, 20 \rangle$ and:
$$\text{HEAP}[20] = f/2.$$
3. deref(X_3) reaches $\langle \text{STR}, 20 \rangle$ and:
$$\text{HEAP}[20] = g/2.$$
4. deref(X_3) reaches a structure representing:
$$a/0.$$

Explain the Machine

Answer each in one or two sentences.

1. Why are query terms built bottom-up?
2. Why are program terms processed top-down?
3. Why does `unify_value` call the general `unify` operation in read mode?
4. Why does $\mathcal{M}_1$ remove predicate names from the heap?
5. Why is the A_i notation useful even though A_i is not a new kind of register?

Key Changes

	$\mathcal{L}_1/\mathcal{M}_1$	$\mathcal{L}_2/\mathcal{M}_2$
Program	A set of first-order atoms, with at most one fact per predicate name.	A set of procedure-defining clauses, with at most one clause per predicate name. Clauses may now be facts or rules.
Clause body	Facts only, no body.	Rules may have a body consisting of a sequence of goals.
Query	A single predicate call.	A sequence of goals executed from left to right.
Continuation	`call p/n` transfers control to a procedure, `proceed` terminates a fact.	A continuation register CP records the instruction to resume after a procedure call. `proceed` sets $P \leftarrow CP$.
Variables	Variables are kept in argument and temporary X_i registers.	Variables needed across procedure calls are *permanent* and stored as Y_i variables; others remain temporary X_i variables.
Environment	No environment stack is required.	Rules use an environment frame on `STACK` to preserve permanent variables and continuation information.
Rule entry/exit	Not applicable.	Use `allocate N` on entry to a rule and `deallocate` when leaving it.

8 Rules and Permanent Variables

Exercise 8.1 : Classify the Variables

Consider:

$$p(X, Y) :- q(X, Z), r(Z, Y).$$

For each variable, state whether it is **temporary** or **permanent**, and give its register allocation.

Variable	Temporary/Permanent	Register
X	_____	_____
Y	_____	_____
Z	_____	_____

Note: The head atom is considered to be part of the first body goal.

Exercise 8.2 : Compile an M_2 Rule

Complete the M_2 code for:

$$p(X, Y) :- q(X, Z), r(Z, Y).$$

```
p/2:
    allocate ______

    get_variable ______, A1
    get_variable ______, A2

    put_value    ______, A1
    put_variable ______, A2
    call q/2

    put_value ______, A1
    put_value ______, A2
    call r/2

    deallocate
```

Answer briefly:

1. Why is Z introduced using `put_variable`?
2. Why are both arguments of `r/2` prepared using `put_value`?

9 Compile and Trace M_2

Exercise 9.1 : Facts and Query

Using the rule above, also consider:

$$q(a, b).$$

$$r(b, c).$$

and the query:

$$?- p(U, V).$$

Complete the code.
Fact `q(a,b).`

```
q/2:
    get_structure ______, A1
    get_structure ______, A2
    proceed
```

Fact `r(b,c).`

```
r/2:
    get_structure ______, A1
    get_structure ______, A2
    proceed
```

Query `?- p(U,V).`

```
    allocate 0
    put_variable ______, A1
    put_variable ______, A2
    call p/2
    deallocate
```

Exercise 9.2 : Trace the Resolution

Trace:

$$? - \ p(U, V)$$

using:

$$p(X, Y) : - \ q(X, Z), r(Z, Y),$$

$$q(a, b), \qquad r(b, c).$$

Fill in the substitutions.

$$p(U, V) \quad \text{with} \quad p(X, Y)$$

gives:

$$X = _____, \qquad Y = _____.$$

Hence:

$$? - \ q(U, Z), r(Z, V).$$

Next:

$$q(U, Z) \quad \text{with} \quad q(a, b)$$

gives:

$$U = _____, \qquad Z = _____.$$

The remaining goal is:

$$? - \ r(_____, V).$$

Finally:

$$r(b, V) \quad \text{with} \quad r(b, c)$$

gives:

$$V = _____.$$

Therefore:

$$\boxed{U = _____, \qquad V = _____}$$