

12 Optimising Compilers (tmj32)

We wish to use abstract interpretation to analyse the construction, modification and traversal of directed graphs. Graph nodes are represented by the following C-like structure; the root of a graph is a pointer to a `graph_node`:

```
typedef struct graph_node {
    int value;
    struct graph_node *children;
} graph_node;
```

Graph nodes are assumed to have a maximum of two children. Exceptions (for example, caused by trying to add a third child to any node, or a search failing to find a node) cause control to transfer out of the program, and do not need to be considered further in the analysis.

- (a) The first analysis consists of identifying whether a graph is actually a tree.
- (i) Create a three-value abstraction for this analysis, describing abstract values and the concrete values that they represent, and why it is safe. [4 marks]
 - (ii) Define the abstract interpretation of the following concrete functions giving a brief explanation for each.
 - (A) Function *create_child*(*g*) creates a new `graph_node` and makes it a child of *g*, returning the new child. [2 marks]
 - (B) Function *add_child*(*g*, *c*) makes an existing node, *c*, a child of *g*, returning *g*. [3 marks]
 - (C) Function *remove_child*(*g*, *c*) removes node *c* from *g*'s children, returning *g*. [2 marks]
 - (D) Function *dfs*(*g*, *v*) locates and returns the first node in a depth-first search starting at *g* that contains the value *v*. [2 marks]
- (b) The second analysis consists of calculating the length of the shortest path from a node to any leaf node. Create an abstraction for this analysis and define the abstract interpretation for the four functions in Part (a)(ii). [*Hint*: consider using a tuple for your abstract values.] [7 marks]