Foundations 2 **Heriot Watt University**
Calum Gilchrist

Foundations 2 Assignment: Part 1 27. Januar 2013

1 Files

1.1 all.h

Contains all the structure definitions. Due to recursive dependencies the structures are all defined together.

1.2 constants.h

Constants required for determining the type of data stored in a container. Ideally these should be stored as enum, but instead are constant definitions.

1.3 value.c

The Value structure is the main storage container used. The Value stores either an Integer, Set or a Pair inside a Container Union. The Value also stores a type value, which is defined in `constants.h`.

1.3.1 Methods

`create_empty_value` function. This function takes a type and creates a new Value with the type field set to that of the type given in arguments. The function then returns the new Value.

`value_equality` Uses the underlying Value.type to determine the equality, using the type_print method, defined by that structure.

1.4 pair.c

The Pair struct has two Value child elements: left and right.

1.4.1 Methods

The `create_pair` method creates a new Pair, with the left and right branches assigned and returns the Value representation of this new pair.

`pair_equality` compares each side of the pair with the corresponding side of the second pair. Will not work for different ordered pairs.

1.5 set.c

The Set structure is implemented as a linked list, so every member of the set knows about it's following member, until you get to the end of the set. A member in the list is a node with a next element and a stored Value.

The `create_set` function creates a new Set in memory and returns the address inside the set_new argument.

The `insert_el` function adds new elements onto the end of the set. The set is recursed through and when the end is found, a new element with the Value of key is created.

`set_contents_equality` compares the contents of two sets. Every element in first is compared with the element in the same position in second. This means that unordered sets are not dealt with. Two sets are not equal if the function finds any element pair that are not equal in the respective sets, before reaching the end of the set.

1.5.1 Set Methods

The Set structure also has some set specific functions for manipulating the sets.

- Union - The union appends every element of the second set onto the end of the first set.
- Subtraction (Complements) of two sets - finds the elements in the first set that are also in the second set and removes them from the first set.
- Intersection - Searches through the sets finding equal values and builds a new set from all the elements that are equal to one another.

1.6 z-notation-parser.c

The main file. Builds the sets required for the assignment and outputs them to the specification.