

This document illustrates the pattern transformation using Z-notation.

$$PATTERNS := \{\text{While Loop Pattern, While loop Incr Pattern, While loop Decr Pattern, Append Array Pattern}\}$$

While Loop Pattern.

let WyIL code block

pattern? : PATTERNS

$$while_loop() := \text{if WyIL code block contains A while-loop} \\ \text{then } true \text{ else } false;$$
$$loop_var(v) := \mathbf{if}(v \neq NULL) \mathbf{then} \textit{true} \mathbf{else} \textit{false};$$
$$\text{loop_init}(v, \text{init}) := \text{if}(v \neq \text{NULL} \wedge \text{init} \neq \text{NULL}) \text{ then } \text{true} \text{ else } \text{false};$$
$$\text{loop_cond}(v, op, b) := \text{if}(v \neq \text{NULL}) \wedge (op \neq \text{NULL}) \wedge (b \neq \text{NULL}) \\ \text{then } \text{true} \text{ else } \text{false};$$

$pattern' =$ While Loop Pattern

While Loop Decrement Pattern

let WyIL code block

pattern? : *PATTERNS*

$$loop_decr(v, 1) := \text{if}(v - -) \text{ then } true \text{ else } false;$$
$$\exists v, init, op, b \bullet (while_loop() \wedge loop_var(v) \wedge loop_init(v, init) \wedge loop_cond(v, op, b) \wedge loop_decr(v, 1))$$
$$\Rightarrow loop_iters(v) == \text{if } op \text{ is } > \text{ then } (init - b) \text{ else } (init - b + 1);$$

$pattern' = \text{While Loop Decr Pattern}$

While Loop Increment Pattern

let WyIL code block

pattern? : PATTERNS

$$loop_incr(v, 1) := \mathbf{if}(v++) \mathbf{then} \textit{true} \mathbf{else} \textit{false};$$
$$\exists v, init, op, b \bullet (while_loop() \wedge loop_var(v) \wedge loop_init(v, init) \wedge loop_cond(v, op, b) \wedge loop_incr(v, 1))$$
$$\Rightarrow \text{loop_iters}(v) == \text{if } op \text{ is } < \text{ then } (b - init) \text{ else } (b - init + 1);$$

$pattern' = \text{While Loop Incr Pattern}$

Append Array Pattern

let WyIL code block
pattern? : *PATTERNS*

```

array_var(arr) := if arr ≠ NULL then true else false ;
array_init(arr, init) := if(init == ∅) then true else false ;
array_append(arr, num) := if(num > 0) then true else false;
∃ v, arr, num • ((loop_incr(v) ∨ loop_decr(v)) ∧ array_var(arr)
                  ∧ array_init(arr, ∅) ∧ array_append(arr, num))
⇒ arr_capacity(arr) == loop_iters(v) * num;
pattern' = Append Array Pattern

```

```

input = input array;
output = output array;
(loop_var(i) ∧ loop_init(i, 0) ∧ loop_cond(i, <, | input |)
  ∧ (loop_incr(i, 1) ∨ loop_decr(i, 1)) ∧ array_var(input)
  ∧ array_init(arr, ∅) ∧ array_append(input, 2))
  ⇒ output_capacity == 2 * | input |

```

PatternTransform

let WyIL code block
pattern? : *PATTERNS*

```

Si=1...4 = a list of statements split by array append pattern;
pattern == pre Append Array Pattern ==  $\exists input, output \bullet ($ 
    S1; output :=  $\emptyset$ ; S2;
    i = 0;
    while(i < | input |){
        S3;
        output := append(output, item1);
        output := append(output, item2);
        S4;
        i ++;
    }
 $\Rightarrow pattern' == \mathbf{pre}$  Resize Array Pattern ==  $\exists input, output \bullet ($ 
    S1; output := {0, 0, ..}; | output | := 2 * | input |; size := 0; S2;
    i := 0;
    while(i < | input |){
        S3;
        output[size] := item1; size ++;
        output[size] := item2; size ++;
        S4;
        i ++;
    }
    assert size <= | output |
    output := resize(output, size);

```
