A Correspondence between the paper and the mechanization

The following table presents a one-to-one correspondence between the paper and the code included in the supplementary material.

Paper Definition	File	Rocq Name
Section 2		
Figure 1	`Semantics/Regex.v`	`regex, anchor, lookaround`
Figure 1	`Semantics/Chars.v`	`char_descr`
Section 3		
Figure 2	`Semantics/Examples.v`	`fig2_tree`
Figure 3	`Semantics/Tree.v`	`tree`
Figure 4, $(l, i, gm, d) \Downarrow t$	`Semantics/Semantics.v`	`is_tree` $l\ i\ gm\ d\ t$
List of actions l	`Semantics/Semantics.v`	`actions`
Input i (zipper)	`Semantics/Chars.v`	`input`
$i_2 <_d i_1$	`Semantics/StrictSuffix.v`	`strict_suffix` $i_1\ i_2\ d$
$\mathrm{idx}(i)$	`Semantics/Chars.v`	`idx` i
$\mathrm{next}(i_{check})_d$	`Semantics/Chars.v`	`advance_input` $i_{check}\ d$
Group map gm	`Semantics/Groups.v`	`GroupMap.t`
$\mathrm{GM}_\emptyset$	`Semantics/Groups.v`	`GroupMap.empty`
$\mathrm{GM}_{\mathrm{open}}(gm, g, n)$	`Semantics/Groups.v`	`GroupMap.open` $n\ g\ gm$
$\mathrm{GM}_{\mathrm{close}}(gm, g, n)$	`Semantics/Groups.v`	`GroupMap.close` $n\ g\ gm$
$\mathrm{GM}_{\mathrm{reset}}(gm, gl)$	`Semantics/Groups.v`	`GroupMap.reset` $gl\ gm$
$\mathrm{advance}(cd, i, d)$	`Semantics/Chars.v`	`read_char` $cd\ i\ d$
$\mathrm{check\_anchor}(a, i)$	`Semantics/Semantics.v`	`anchor_satisfied` $a\ i$
$\mathrm{advance\_backref}(gm, g, i, d)$	`Semantics/Semantics.v`	`read_backref` $gm\ g\ i\ d$
$\mathcal{G}(r)$	`Semantics/Regex.v`	`def_groups` r
$\mathrm{lk\_result}(lk, t_{look}, gm, i)$	`Semantics/Semantics.v`	`lk_group_map` $lk\ t_{look}\ gm\ i$
$\mathcal{L}_0(t, i)$	`Semantics/Tree.v`	`first_leaf` $t\ i$
Theorem 1	`Semantics/Semantics.v`	`is_tree_determ`
$\mathcal{T}(l, i, gm, d, n)$	`Semantics/FunctionalSemantics.v`	`compute_tree` $l\ i\ gm\ d\ n$
$\vert\vert l\vert\vert_d^i$	`Semantics/FunctionalSemantics.v`	`actions_fuel` $l\ i\ d$
$\vert\vert r\vert\vert_d^i$	`Semantics/FunctionalSemantics.v`	`regex_fuel` $r\ i\ d$
$\vert i\vert_d$	`Semantics/FunctionalSemantics.v`	`max_iter` $i\ d$
$\mathrm{worst}(lk, i)$	`Semantics/FunctionalSemantics.v`	`worst_input` $i\ d$
$\mathrm{dir}(lk)$	`Semantics/Regex.v`	`lk_dir` lk
Theorem 2	`Semantics/FunctionalSemantics.v`	`functional_terminates`
$\mathcal{T}(l, i, gm, d)$	`Semantics/FunctionalUtils.v`	`compute_tr` $l\ i\ gm\ d$
Theorem 3	`Semantics/ComputeIsTree.v`	`compute_is_tree`
Section 4		
$\downharpoonleft r_w \downharpoonright$	`WarblreEquiv/RegexpTranslation.v`	`warblre_to_linden`
$\upharpoonleft res \upharpoonright$	`WarblreEquiv/ResultTranslation.v`	`to_MatchState`
Theorem 4	`WarblreEquiv/EquivMain.v`	`equiv_main_reconstruct`
Equivalence relation between continuations and lists of actions	`WarblreEquiv/EquivDef.v`	`equiv_cont`
Section 5		
$r_1 \approx r_2$	`Rewriting/Equivalence.v`	`observ_equiv`

Regex contexts C	Rewriting/Equivalence.v	regex_ctx
$C[r]$	Rewriting/Equivalence.v	plug_ctx C r
Type of context C	Rewriting/Equivalence.v	ctx_dir C
$\mathcal{L}(t, i, d)$	Semantics/Tree.v	tree_leaves t $\mathrm{GM}_\emptyset$ i d
$\ell_1 \equiv \ell_2$	Rewriting/LeavesEquivalence.v	leaves_equiv [] ℓ_1 ℓ_2
$r_1 \sim_d r_2$	Rewriting/Equivalence.v	tree_equiv_dir
$r_1 \sim_\leftrightarrow r_2$	Rewriting/Equivalence.v	tree_equiv
Theorem 5	Rewriting/Equivalence.v	regex_equiv_ctx_samedir
Theorem 6	Rewriting/Equivalence.v	regex_equiv_ctx_forward
Theorem 7	Rewriting/Equivalence.v	regex_equiv_ctx_backward
Theorem 8	Rewriting/Equivalence.v	observe_equivalence
Figure 6	Semantics/Example.v	different_results
Figure 5:		
$r_1 \mid \langle r_2 \mid r_3 \rangle \sim_\leftrightarrow \langle r_1 \mid r_2 \rangle \mid r_3$	Rewriting/Associativity.v	disj_assoc
$r_1 \langle r_2 r_3 \rangle \sim_\leftrightarrow \langle r_1 r_2 \rangle r_3$	Rewriting/Associativity.v	seq_assoc
$\langle r_2 \mid r_3 \rangle r_1 \sim_\rightarrow \langle r_2 r_1 \rangle \mid \langle r_3 r_1 \rangle$ when r_1 has no group	Rewriting/Distributivity.v	factored_expanded_right_equiv
$r_1 \langle r_2 \mid r_3 \rangle \sim_\leftarrow \langle r_1 r_2 \rangle \mid \langle r_1 r_3 \rangle$ when r_1 has no group	Rewriting/Distributivity.v	factored_expanded_left_equiv
Anchors as lookarounds	Rewriting/Anchors.v	desugar_anchor_correct
$\odot$	Semantics/Chars.v	CdAll
$r\{min, 0, \top\} \sim_\leftrightarrow r\{min, 0, \bot\}$	Rewriting/ForcedQuant.v	forced_equiv
Figure 7:		
$r\{min_1, 0, p\} r\{min_2, 0, p\} \sim_\leftrightarrow r\{min_1 + min_2, 0, p\}$	Rewriting/RegexpTree.v	bounded_bounded_equiv
$r\{min_1, 0, p\} r\{0, \Delta_2, \top\} \sim_\rightarrow r\{min_1, \Delta_2, \top\}$	Rewriting/RegexpTree.v	bounded_atmost_equiv
$r\{min_1, 0, p\} r\{0, \Delta_2, \bot\} \sim_\rightarrow r\{min_1, \Delta_2, \bot\}$	Rewriting/RegexpTree.v	bounded_atmost_lazy_equiv
$r\{0, \Delta_1, \top\} r\{min_2, 0, p\} \sim_\leftarrow r\{min_2, \Delta_1, \top\}$	Rewriting/RegexpTree.v	atmost_bounded_equiv
$r\{0, \Delta_1, \bot\} r\{min_2, 0, p\} \sim_\leftarrow r\{min_2, \Delta_1, \bot\}$	Rewriting/RegexpTree.v	atmost_bounded_lazy_equiv
$r\{0, \Delta_1, \top\} r\{0, \Delta_2, \top\} \sim_\leftrightarrow r\{0, \Delta_1 + \Delta_2, \top\}$	Rewriting/RegexpTree.v	atmost_atmost_equiv
Chain of forward equivalences	Rewriting/Chain.v	equivalence_chain
Section 6		
Figure 9	Engine/PikeSubset.v	pike_regex
Subset of actions	Engine/PikeSubset.v	pike_action
Figure 10	Engine/NFA.v	bytecode, code
Figure 11	Engine/NFA.v	compile
Label l	Engine/NFA.v	label
Accept instruction appended	Engine/NFA.v	compilation
Thread (pc, gm, b)	Engine/PikeVM.v	thread
Figure 12	Engine/PikeVM.v	pike_vm_step
States of PikeVM	Engine/PikeVM.v	pike_vm_state
$\mathrm{VM}_{\mathrm{init}}(i)$	Engine/PikeVM.v	pike_vm_initial_state

Figure 15	Engine/FunctionalPikeVM.v	paper_regex
Figure 15a	Engine/FunctionalPikeVM.v	paper_bytecode
Figure 15b	Engine/FunctionalPikeVM.v	paper_tree
Figure 13	Engine/BooleanSemantics.v	bool_tree
Figure 14, $(l, i) \vdash b$	Engine/BooleanSemantics.v	bool_encoding b i l
Theorem 9	Engine/BooleanSemantics.v	encode_equal
Theorem 10	Engine/BooleanSemantics.v	booltree_istree_equiv
Figure 16	Engine/PikeTree.v	pike_tree_step
States of PikeTree	Engine/PikeTree.v	pike_tree_state
$\mathsf{PT}_{\mathsf{init}}(t, i)$	Engine/PikeTree.v	pike_tree_initial_state
$pts \downdownarrows res$	Engine/PikeTree.v	piketreeinv pts res
$(t, gm) \downarrow^{i}_{\mathcal{S}} res$	Engine/PikeTree.v	tree_nd t gm i $\mathcal{S}$ res
$\mathcal{A} \downarrow^{i}_{\mathcal{S}} res$	Engine/PikeTree.v	list_nd $\mathcal{A}$ i $\mathcal{S}$ res
$(i, best, \mathcal{A}, \mathcal{B}, \mathcal{S}) \downarrow res$	Engine/PikeTree.v	state_nd i $\mathcal{A}$ $best$ $\mathcal{B}$ $\mathcal{S}$ res
Theorem 11	Engine/PikeTree.v	init_piketree_inv
Theorem 12	Engine/PikeTree.v	pts_preservation
Figure 17	Engine/PikeEquiv.v	tree_thread
$rep_{code}\ l\ pc$	Engine/NFA.v	actions_rep
$\mathcal{A} \sim^{i}_{code} \mathcal{A}'$	Engine/PikeEquiv.v	list_tree_thread
$\mathcal{S}_{VM} \subseteq_{\mathcal{A}} \mathcal{S}$	Engine/PikeEquiv.v	seen_inclusion
Figure 18	Engine/PikeEquiv.v	pike_inv
Theorem 13	Engine/PikeEquiv.v	initial_pike_inv
Theorem 14	Engine/PikeEquiv.v	invariant_preservation
Theorem 15	Engine/Correctness.v	pike_vm_to_pike_tree
Theorem 16	Engine/Correctness.v	pikevm_warblre