Current Implementation

Runs after type checking, before IR generation.

Currently very buggy:

- compiler: Function return values (?) not life-time-tracked #1497

  Long standing issue, fix is simple but is not yet fixed because global lifetime needs special handling.

- Unsound parameter lifetime handling #1677

  Issue found today after reading the code.

- List elements not lifetime-tracked #1678

  Also found today... This also affects object with fields.

The later 2 bugs probably require algorithmic changes, simple fixes are not sufficient.

Ideas:

1. Three ways for a pointer to escape:
   - Assigning it to another object.
     Solution: Ensures outlive relationship.
   - Returning it to caller.
     Solution: Only allow returning values that live forever.
   - Raising an exception containing the pointer.
     Solution: Forbid allocating exceptions that refer to mutable data.
2. The current algorithm check the lexical scope of variables, validate the outlive relationship for assignments.
3. For expressions like the if expressions which may be evaluated to multiple objects, it would choose the shortest living operand.

Problems:

1. The lifetime of different parameters should be incomparable instead of the same.
2. The current algorithm would not track fields/list elements.
3. Choosing the shortest living operand is unsound:
   - Variable a Lifetime: Global / Local (selected with if expression).
   - Variable b Lifetime: Local.
   - By choosing shortest living operand, lifetime of a is considered as local.
   - $a.foo \leftarrow b$ is now legal as b outlives local lifetime.

Conclusion: We probably need a completely new algorithm.

Analysis approach?

We track the (transitive closure of?) points-to set of each variable, classify the pointee into local L, nonlocal N and global G with the following relations:

1. Global outlives global, nonlocal and local variables, and can be returned. $G \geq G, N, L$.

   Primitive immutable types and cache data have this lifetime.

2. Nonlocal variables outlives local variables, but not other nonlocal variables. This can be returned. $N \geq L$.

   Function parameters have nonlocal lifetime.

3. Local variables outlive other local variables. $L \geq L$.

We require assignments RHS outlives the LHS. More specifically, we require min of RHS outlives max of LHS. For example, if RHS can point to local variables, LHS must not point to variables that are nonlocal or global. If RHS can only point to global variables, LHS can point to whatever variables.

For return, we require the points-to set cannot contain local variables.