

## Environment Diagrams

### *Formal rules for the environment diagram*

- To evaluate a compound expression (other than a special form), first evaluate the subexpressions. Then apply the value of the first to the rest.
- The value of a variable with respect to an environment is the value given by the binding of the variable in the first frame that contains such a binding.
- Evaluating a lambda expression produces a procedure object, which consists of two parts. The first is the code of the procedure, which is given by the text of the lambda expression. The second is the environment pointer, which points to the environment in which the lambda expression was evaluated to produce the procedure.
- Define adds a binding to a frame.
- A procedure object is applied to a set of arguments by constructing a frame, binding the formal parameters of the procedure to the actual arguments of the call, and then evaluating the body of the procedure in the context of the new environment. The new frame has as its enclosing environment the environment part of the procedure object being applied.

### *Simplified instructions*

Mastering environment diagrams is simply a matter of learning the rules. The formal rules of the model are given above. Here are simplified instructions for you to follow that capture what the rules are telling you to do.

### *What are environment diagrams made of?*

- Frames We draw a frame as a box. In the box go bindings. Every frame (except the frame representing the global environment) needs to have a link to exactly one other frame.
- Bindings A binding is an association between an identifier (or symbol) to a value. If the value is a list, we draw it as a box and pointer diagram. If the value is a procedure, the binding is linked to a procedure object. If the value is a number, we write something like a:3 in the frame, where a is the identifier we are binding and 3 is the value a is bound to.

- Links or Pointers A link (or pointer) is a connection from one frame to another. We look up the chain of frames using these links when looking up the value of an identifier.
- Procedures These are special objects created by a lambda expression as described below.

### *Define*

- Define adds a binding to the frame in which it is evaluated.

### *Looking up an identifier*

- Look for a value in the current frame.
- If there is no value for that identifier in the current frame, follow the link from the current frame to the one that it is hung from.
- Continue in this manner until we find a binding for the identifier we're looking up or until we run out of frames in our chain of links (i.e. we come to the global environment). If there's no binding in the global environment, the identifier we're looking up is an unbound variable.

### *Lambda*

- Evaluating a lambda expression will result in a two-part procedure object (two circles next to each other).
- The pointer of the left circle points down to a list of the parameters and the body of the procedure.
- The pointer of the right circle points up to the environment frame in which the lambda expression was evaluated.
- Note that nothing else happens until we apply the procedure.

### *Applying a procedure*

To apply a procedure object, follow these three steps:

- Draw a new environment frame. Bind the parameters to their values in this new environment frame.
- Link the new environment frame to the environment frame pointed to by the right circle of the procedure object.
- Evaluate the body of the procedure in this new environment frame.