Notes on an implementation of Heim 1983

Chris Barker, 31 Jan, github.com/cb125/Dynamics

1. Heim 1983 sketches a dynamic grammar that models presupposition projection and some anaphora.
2. **Contexts**:
   - Sentences are evaluated with respect to a context
   - a context is a set of evaluation points
   - each evaluation point is an ordered pair consisting of an assignment function and a world
3. **Context Change Potential**:
   - sentences denote a CCP, which is a function from a context to an updated context.
   - if a sentence is evaluated with respect to a context that does not satisfy the presuppositions of the sentence, the context is not admissible, and there is a presupposition violation. So the result of updating a context with a sentence is either a set of evaluation points, or presupposition violation.
4. **Update is eliminative**: the updated context is always a subset of the input context
5. **Assignment functions** for Heim are total functions from the natural numbers into the set of individuals.
   - Assignment functions can therefore be represented by a sequence of individuals
   - The ith element in the sequence g is the value of g(i)
6. In the implementation here, the discourse is limited to talking about the first five natural numbers, namely, 1, 2, 3, 4, and 5.
   - So assignment functions are sequences of integers, e.g., [2,3,2]
   - Many predicates have their normal meaning: *even*, *odd*, *prime*, etc.
   - One quirk is that 5 has no successor in the domain of discourse, so expressions like "5's successor" will give rise to a presupposition failure
7. **The novelty condition**: Some lexical items imposes a constraint on contexts, her (22). A context c satisfies this constraint just in case

   (22) For any two sequences g and g' that differ at most in their ith member, and for any world w: $(g, w) \in c$ iff $(g', w) \in c$.

   As Heim explains, this constraint guarantees that the index i is "unused", and that every world in the context will be associated with assignment functions that map the index in question onto every individual in the domain of discourse.

This constraint is used in the definition of the CCP of *every* and of *a(n)*. With respect to *a(n)*, it corresponds to Heim's 1982 novelty constraint.

8. The novelty condition is considered to be part of the indefinite determiner's presuppositions, a presupposition about the state of the scoreboard.
9. Because novetly is a property of contexts (not of individual evaluation points), and because we must guarantee novelty for indefinites embedded arbitrarily deeply, it is necessary to pass the entire evaluation context throughout the composition.
10. As in a number of later dynamic approaches, it is not essential to test for novelty by examining the context set. Instead, it is possible to augment evaluation points with a record of the highest-used index. Then evaluating an indefinite simply uses the next higher index, populates that index with the full range of suitable individuals, and increments the record for the sake of downstream indefinites.
    - With this refactoring, Heim's system will (probably, need to double check) be **pointwise**. That is, the update of a set of points will always be equal to the union of the update of singleton contexts containing each of the points of the original context. That is,

    $$c + A = \bigcap_{c_i \in c} \{c_i\} + A$$

    What Gronendijk and Stokhof 1990/2005 call "**distributivity**".
11. Clauses denote context update functions.
    - Heim explicitly gives recipes for computing some clauses based on the meaning of their subparts. For instance, she gives an update for sentences of the form "If A B", where A and B are clauses in their own right.
    - In the recipe for "every" and "a(n)", determiners combine with a restriction A and a nuclear scope B, which are turned into clauses according to the following schema:
      - Every Pred VP = Every x: (Pred x) (VP x)
    - Although we find out how variables depend on assignment functions, Heim does not specify the denotations of smaller expression types such as adjectives, nouns, or verbs.
12. In the implementation worlds are indexed by integers. So the Haskell type of a context is [([Int], Int)]: a set of pairs consisting of a list of integers (the assignment function) and an integer (the world).
13. There is exactly one predicate in the fragment that is world-sensitive: *allowable*. For any world w, and integer i counts as allowable in w just in case $i < w$. So in world 2, the integers 3, 4 and 5 are allowable.

Examples from running the implementation

```
(3 is allowable): Just [([],4),([],5)]
```

This example shows that the sentence “3 is allowable” is true with respect to worlds 4 and 5, but not with respect to worlds 1, 2, or 3. There are no quantifiers or variables, so we can leave the assignment functions empty.

Heim pretends for simplicity that “if” is just material implication.

```
(if (2 is allowable) (3 is allowable)): Just [([],1),([],2),([],4),([],5)]
```

This sentence is not true with respect to world 3, since in that world, 2 is allowable, but 3 is not. This is not a good theory of the meaning of natural language conditionals, but a more sophisticated theory of the conditional would be orthogonal to the concerns of presupposition projection and anaphora.

(It would be fairly easy to implement a variable strict theory of conditions by treating world similarity as arithmetic difference, so that worlds 2 and 3 count as more similar than 2 and 4.)

As mentioned, evaluating *5's successor* results in presupposition failure:

```
(2's successor is odd): Just [([],1),([],2),([],3),([],4),([],5)]
(5's successor is odd): Nothing
```

Here, “Nothing” is how the implementation says “presupposition failure”.

```
(if (it_1 is prime) (it_1 is even)):
Just [([1],1),([1],2),([1],3),([1],4),([1],5),
      ([2],1),([2],2),([2],3),([2],4),([2],5),
      ([4],1),([4],2),([4],3),([4],4),([4],5)]
```

This sentence is true in every world, but only for assignment functions that map the first index to 1, 2, or 4.

Here is a variable in action:

(every_1 allowable prime is odd): Just [([1],1),([1],2),([2],1),([2],2),([3],1),([3],2),([4],1),([4],2),([5],1),([5],2)]

This sentence is true only in worlds where 2 is not allowable, namely, worlds 1 and 2.

Indefinites:

```
(a_1 prime is even): Just [([2],1),([2],2),([2],3),([2],4),([2],5)]
```

A point survives update with the first sentence only if it maps index 1 to the only even prime, namely, 2.

```
(a_1 prime is odd): Just [([3],1),([3],2),([3],3),([3],4),([3],5),
                          ([5],1),([5],2),([5],3),([5],4),([5],5)]
```

A point survives update with the second sentence if it maps index 1 to either 3 or 5.

```
((a_1 prime is even) and (it_1's successor is odd)):
Just [([2],1),([2],2),([2],3),([2],4),([2],5)]
```

Update with the first sentence restricts the context to assignments that map the index 1 to the even prime.

```
((a_1 prime is odd) and (it_1's successor is even)): Nothing
```

This example illustrates the discussion in the final section before the conclusion. The first conjunct restricts the context to assignment functions that map index 1 to either 3 or 5. The second sentence presupposes that the referent of the pronoun has a successor. In this model, 3 has a successor, but 5 does not. According to Heim's fragment, the fact that 5 doesn't have a successor makes the entire sentence fail. Heim considers this to be a flaw in the theory. This phenomenon is poorly understood; discussed in my 95 dissertation,

– add example showing that left conjunct shadows presuppositions and that antecedents shadow consequents.