Supplementary Notes on Ch. 2 of McDermott: Language ***CSC 191/291***

Ch.2 of McDermott provides a good selective survey of some important AI techniques. He is surely correct in arguing that humans (and no doubt other animals) rely on numerous specialized capabilities in their perceptual, motor, and cognitive functioning. However, his conclusion that there is no "language of the mind", no general language understanding principles, or no general inference methods, relies too much on consideration of the distinctive specialized perceptual and motor processes needed by any agent functioning in the world. His conclusion is not one that would be accepted by most AI researchers who actually try to implement natural language understanding capabilities and inference capabilities in machines. McDermott's view is essentially the (original) "MIT view", strongly influenced by Minsky -- that the mind is a collection of a large number of complex "specialist" modules, whose interactions somehow give an illusion of unity.

> [***Caveat***: McDermott's negative view of a general mental knowledge representation and reasoning is currently quite common and in my view deleterious for AI, one that has handicapped progress towards human-level AI for a quarter century. Amazingly, John McCarthy, co-founder of AI (along with Marvin Minsky, Alan Newell, and Herbert Simon) and most famous exponent of the use of logical knowledge representations in AI, is mentioned neither in ch.2 nor elsewhere in the book. As I will reiterate, what has been neglected is the semantic uniformity of all human languages -- all possess the same set of semantic expressive devices of predication, and/ or/ not/ if-then, quantification, modification, etc. (see again below), even though on the surface they can look structurally quite different. This semantic uniformity virtually shouts for explanation -- and the most obvious explanation is that the same types of semantic resources exist in the human brain. It may well be that this is what gave rise the exceptional capacity of homo sapiens for both language and thought circa 200-300 thousand years ago. And it is often forgotten that logics, at least in their semantic expressivity, and in their entailment relations, are derivative from language, rather than being some sort of esoteric invention by mathematically minded thinkers.]

Natural Language Understanding (NLU)

Here is a quick overview of what natural language understanding involves, rather different from McD's view of all intelligence as special-purpose routines:

First, if we start with spoken language, the acoustic signal needs to be mapped to a sequence of words (or a probabilistic lattice of possible word sequences). This involves analyzing the frequency spectrum, and mapping this in stages to phones, phonemes, and then words. Phones are the distinct speech sounds that occur in human languages. Phonemes are groups of phones that are equivalent in a given language, in terms of what they contribute to the structure of a (spoken) word; e.g., in English "depict" might have two short "i" sounds, both perceived as such but actually corresponding to different phones. Without going into details, the various processing stages typically use *generative* models, involving state-to-state transitions where each state probabilistically generates an observable element. If the current state of the model is influenced by only one or two earlier states -- there is no "memory" of what happened earlier in the sequence -- we have a "Markov" model (in fact a "hidden Markov model", or HMM, assuming that the states themselves are not observable). For example certain frequency spectrum features over brief

speech segments, computed by signal processing methods, might be viewed as the observable elements generated by a Markov model whose states correspond to ("hidden") phonemes, to be inferred from the observed phones. Once a likely phoneme sequence has been inferred (using e.g., the Viterbi algorithm, a fast dynamic programming method), a similar kind of HMM can be used to infer a likely word sequence, where the words are thought of as generating several phonemes in a row. (For phoneme-to-grapheme inference, see the 44 phonemes at https://www.dyslexia-reading-well.com/44-phonemes-in-english.html .)

So let's assume we have word sequence like *"Romeo loves Juliet"*. How do we get to an internal representation that can be used for inference? (E.g., if we also have a general fact that if x loves y, then x wants to be near y, we should be able to infer that Romeo wants to be near Juliet; or, *"Romeo loves Juliet"* might itself be an inference, if all we know is that *"Everyone (i.e., every person) loves Juliet" and "Romeo is a person".*) Very sketchily, what we do is to *parse* the sentence, assigning it a hierarchical (tree-like) structure, and then use rules that tell us how to map that structure to a logical representation. Here's a parse tree for our sentence, and the rules it is based on:

```
      S                 Phrasal rules:     Semantic (logical form) rules:
    /   \               VP --> V  NP       VP' = (V' NP')        The rules "compose" the parts,
   NP   VP              S  --> NP  VP      S' = (VP' NP')        with lambda-conversion where
   |     | \            Lexicon:                                 possible
Romeo   V  NP           Romeo: NP          NP' = Romeo1
        |   \           Juliet: NP         NP' = Juliet1
     loves Juliet       loves: V           V' = (lambda y (lambda x (love x y)))
```

Here we've recognized *"Romeo"* as a complete noun phrase (NP), *"loves"* as a verb, and *"Juliet"* as another complete noun phrase. (We get these facts from a lexicon for English.) We've also recognized the "V NP" combination as a verb phrase (often called the predicate of a sentence); and finally we've recognized that the NP *"Romeo"* can now combine as subject of the sentence with the predicate VP, *"loves Juliet"*, forming a sentence, S. In general there will be ambiguities in this recognition process, but we set this aside for now.

Next we perform "bottom-up" semantic interpretation, i.e., mapping the parse tree to a logical formula. Here we use a semantic lexicon and rules for interpreting each phrase type. The semantic lexicon tells us that the logical name (constant) corresponding to *"Romeo"* is (say) Romeo1, and the one for *"Juliet"* is Juliet1. For "*loves",* the semantic entry is a bit trickier, as you see above:

(lambda y (lambda x (love x y))). (*In Lisp we would write* (lambda (y) (lambda (x) (love x y)).)

As in Lisp, the lambdas just specify the order of argument binding; i.e., y (the outermost lambda-variable) is to be bound first when we apply the lambda expression to an argument, and x is bound after that. We'll see this in action in a moment. So first, we label the nodes NP, V, NP to which *"Romeo", "loves" and "Juliet"* are attached with their semantic (logical) counterparts Romeo1, (lambda y (lambda x (love x y))), and Juliet1 respectively. We now compute the semantic value of the VP using the rule VP' = (V' NP'), where we use primes to refer to the semantic values of parse tree nodes. The result is

VP' = ((lambda y (lambda x (love x y))) Juliet1)
= (lambda x (love x Juliet1)).

Finally we apply the rule S' = (VP' NP') at the top level of the parse tree, obtaining

S' = ((lambda x (love x Juliet1)) Romeo1)
= (love Romeo1 Juliet1).

So that's the resulting "logical" formula. As you see, that's a lot like the original English, but it has been made clear what the logical relation is ('love') and what its first and second arguments are, thanks to the bracketing and ordering of these arguments.

A slightly more complicated case is one where instead of *"Romeo"* in the original sentence we have *"everyone"* (= *"every person"*). Assume that the semantic lexicon supplies the value <every person> for this, a so-called unscoped quantifier. Then you can see that the resulting logical form for the altered parse tree will be

S' = (love <every person> Juliet1).

Now, *"every"* is the same as the universal quantifier (as used, e.g., in mathematics or formal predicate logic -- the upside-down A), and as such should appear *outside* the sentence; for example in number theory we might have the claim

(every x (exists y (y > x)))

(for every number x there is a larger one y). So we still need to move the unscoped quantifier <every person> to the "outside" of our formula. When we do this, we also introduce a variable (bound by the quantifier), and we use the 'person' part of the unscoped quantifier to *restrict* the quantification to persons:

S' = (every x: (person x) (love x Juliet1)).

Note that the colon after the x indicates that the next subformula, here '(person x)', restricts the values of x that we are considering. In other words, we're only considering *persons* x when making the claim '(love x Juliet1)'. In standard predicate logic notation (with Lisp bracketing), this is equivalent to

S' = (every x ((person x) => (love x Juliet1))),

i.e., for every x, if x is a person then x loves Juliet. (However not all natural quantifiers, such as "most", can be represented in standard predicate logic, whereas with the quantifier restriction notation, we can easily say (most x: (person x) (love x Juliet1)).)

So now we can also see how inference would work. For example, given the premises

(every x ((person x) => (love x Juliet1))),
(person Romeo1),

it's intuitively obvious that we can conclude

(love Romeo1 Juliet1),

much as in our previous example of inferring that the dog, Snoopy, has a tail, knowing that every dog has a tail. As mentioned at the time, the formal rule allowing this inference is "UI+MP", i.e., universal instantiation plus modus ponens. The name doesn't matter, but what does matter is that we can *justify* this rule on perfectly general principles -- the conclusion is guaranteed to be true if the premises are true. Showing this requires a theory of truth, which in turn requires denotational semantics – which we've briefly studied, and which McDermott unfortunately rejects. Recall that denotational semantics permits certain correspondences between symbols and entities in the domain we're considering. For example, 'Romeo1' could correspond to a particular boy, 'Juliet1' could correspond to a particular girl, 'person' could correspond to a set of persons, and 'love' could correspond to all pairs of entities where the first loves the second. If

(every x ((person x) => (love x Juliet1)])

is true, then the set of pairs denoted by 'love' must include *all* pairs where the first element of the pair is in the set denoted by 'person', and the second element is the girl denoted by 'Juliet1'. Thus, if the boy denoted by 'Romeo1' is in the set of persons, the pair consisting of that boy and the girl denoted by 'Juliet1' must be in the set denoted by 'love' -- and that's why the conclusion

(love Romeo1 Juliet1)

above is true, as previously noted! The key point is that the argument will go through even if we are talking about numbers, planets, or whatever. For example, imagine that we are talking about celestial objects in our solar system, and 'person' refers to the planets, 'Juliet1' denotes the Sun, 'Romeo1' denotes the Earth, and 'love' denotes the relation of one thing orbiting around another. (Thus we are saying every planet orbits around the Sun.) Then the conclusion (love Romeo1 Juliet1) still follows from the premises, but now it "means" that Earth orbits the Sun! Such inferential power is not to be dismissed lightly ...

McDermott does effectively explain some of the difficulties involved in mapping language to an internal representation, but the reasons he gives for pessimism seem to me on the wrong track. One reason he gives is that understanding a sentence may require finding an interpretation that minimizes contradictions, which is very hard. But that's just not the way language works. For example, we tend to interpret "*He saw a monkey with yellow tail feathers*" as if the monkey sported a bird-like feathery tail, even though we know perfectly well that monkeys are furry, not feathery -- and even though a consistent interpretation is available in which the monkey is HOLDING the detached tail feathers of a bird. Most of the time, interpretation seems based more on familiar patterns of language and predication, than on any consistency or plausibility checks.

Another reason McDermott gives (p.67) is that a symbolic representation is unusable if you can't guarantee that every problem that can be expressed in it can be efficiently solved in it (and of course even boolean satisfiability is intractable unless P=NP...) This is an unfounded and even perverse claim that has hamstrung much "logical" AI research in the last 25 years. Obviously, PEOPLE can comprehend (hence presumably represent to themselves, somehow) problems that they find extremely difficult or even impossible to solve -- Fermat's last theorem waited more than 3 1/2 centuries for a proof, and Goldbach's conjecture, the continuum hypothesis, and the P=NP? question, etc., remain unsolved. So why should AI systems be prevented from even being able to REPRESENT such problems? The arguments that are made (e.g., by Brachman and Levesque in their text, *Knowledge Representation and Reasoning*) to support such a position are very much like saying: beware of using programming languages that allow recursion or looping, because then you can't guarantee fast termination of all problem-solving algorithms that you can implement in that language!

I'd also warn you about McD's discussion of reasoning. On p.68, he argues that (1) people don't do a significant amount of deduction (which I believe is a serious mistake, as I argue in my reply to his "Critique of pure reason", in the same issue of Computational Intelligence as that paper); (2) computers do a lot of nondeductive reasoning such as planning (very true; I would add "schema-based expectation" as major form of inference); (3) both types of reasoning often lead to mistaken conclusions (true, in part because of the "guesswork-like" methods we use and in part because the premises FROM which we, or our computers, reason are often unreliable, being based on unreliable sources.); THEREFORE "computers don't deduce, they calculate". (This is a non-sequitur.)

Commenting on efforts to build large knowledge bases over expressive representations, such as Cyc, he reiterates his skepticism about any general representational and reasoning techniques. He thinks we need to just keep compiling an ever-larger collection of specialized methods. He declares that the intuition people have that we have a comprehensive internal knowledge representation (in particular, for the content of language) is an "illusion" based on introspection -- we are misled by our self-model.

I would counter, coming back to my earlier point, that we have plentiful *external* manifestations of such an internal representation, *namely human languages themselves!* I cannot comprehend why someone who takes a serious look at the structure of human languages and the way they convey meaning could fail to be struck by the following expressive devices shared by ALL human languages:

- naming of individual entities ("Plato", "New York City", or "Hurricane Katrina");
- predication ("Plato is a man"), "The cat is on the mat");
- connectives ("and", "or", "not", "if ... then");
- generalized quantifiers ("all humans", "most Western democracies")
- equality ("Lincoln's assassin was John Wilkes Booth")
- predicate modification ("very smart", "dances gracefully")
- sentence modification ("Perhaps there is life on Europa")
- predicate reification ("happiness", "humankind")
- sentence reification ("The fact that Brutus stabbed Caesar")
- event reference and modification ("Molly barked last night. This went on for an hour, and woke all the neighbors.")

(Two or three further items could be added.) Surely this has some significance -- and as noted the most straightforward assumption we could make is that these features are a reflection of our internal representational capabilities -- which suggests that our "mentalese" is a kind of enriched logic-like, language-like representation, the very hypothesis McDermott scoffs at.