Constructive Logic (15-317), Fall 2017
Recitation 12: Substructural logics, a substandard*recap.

What is a Substructural Logic

The core idea with substructural deductions as was shown in lecture yesterday is that restricting the logic gives us the ability to express new things. Specifically, we got rid of two so called "structural principles". These are the rules of weakening and exchange:

$$\frac{\mathcal{J}_0 \quad \mathcal{J}_1}{\mathcal{J}_0}\ \textit{Weakening} \qquad \frac{\mathcal{J}}{\mathcal{J} \quad \mathcal{J}}\ \textit{Contraction} \qquad \frac{\mathcal{J}_0 \quad \mathcal{J}_1}{\mathcal{J}_1 \quad \mathcal{J}_0}\ \textit{Exchange}$$

By getting rid of weakening and contraction our propositions represent not simply eternally true pieces of knowledge but rather ephemeral truths. This opens the possibility of expressing things that are only temporarily true directly in the logic. For instance, it's rather unnatural to encode currency in normal (structural) logic. After all, knowing that you have one dollar entitles you to conclude that you have two dollars! This is an interesting concept but experience has demonstrated that it is not so. If we sacrifice just weakening, we are working with *linear* logic and if we sacrifice both weakening and exchange we are working in an *ordered* logic.

Binary Numbers

Let us turn first to ordered logic. There's a natural way to encode a binary number in a logic where judgments may not rearrange themselves or suddenly be dropped; we simply turn 01110 into the knowing `b0 b1 b1 b1 b0`. This would never work in a structural or even linear logic (why?). For the sake of convenience, we actually denote the end of a binary number with $. So we would properly encode the above number as

`$ b0 b1 b1 b1 b0`

Note that in this encoding we explicitly don't care about preserving standard form. The least significant bit is on the right and the most signficant bit is on the left but leading zeroes are explicitly allowed for simplicity.

Task 1. Design a proposition `inc` so that when `inc` is placed to the right of a binary number it is possible to derive the successor of that binary number.

*GET IT?!?

$$\frac{\$ \quad \mathsf{inc}}{\$ \quad \mathsf{b1}} \qquad \frac{\mathsf{b0} \quad \mathsf{inc}}{\mathsf{b1}} \qquad \frac{\mathsf{b1} \quad \mathsf{inc}}{\mathsf{inc} \quad \mathsf{b0}}$$

Task 2. Design a proposition dec so that when dec is placed to the right of a binary number it is possible to derive the successor of that binary number. If there is no positive predecessor of the number derive ☹.

$$\frac{\$ \quad \mathsf{dec}}{☹} \qquad \frac{☹ \quad \mathsf{b1}}{☹} \qquad \frac{\mathsf{b0} \quad \mathsf{dec}}{\mathsf{dec} \quad \mathsf{b1}} \qquad \frac{\mathsf{b1} \quad \mathsf{inc}}{\mathsf{b0}}$$

A worthwhile exercise is to modify these programs so that they do expect and preserve standard forms of binary numbers.

Graphs

Moving to a new example, let us consider how we might encode graphs in this framework. Ordered logic is a less natural fit here so we turn instead to linear logic. Let us suppose that we have a number of propositions $\mathsf{node}(a)$ and $\mathsf{edge}(a, b)$ denoting the obvious things. We shall assume that the graph is undirected but it will be ideal if we do not include both $\mathsf{edge}(a, b)$ and $\mathsf{edge}(b, a)$ for every edge. What can we express now with inference?

Task 3. Write a series of inference rules to deduce when there is a path from a to b.

$$\frac{\mathsf{edge}(a, b)}{\mathsf{edge}(b, a)} \qquad \frac{\mathsf{node}(a)}{\mathsf{path}(a, a)} \qquad \frac{\mathsf{path}(a, b) \quad \mathsf{edge}(b, c)}{\mathsf{path}(a, c)} \qquad \frac{\mathsf{edge}(a, b) \quad \mathsf{path}(b, c)}{\mathsf{path}(a, c)}$$

By encoding these substructurally we have prevented ourselves from being able to derive a path that uses the same edge multiple times. This means that all paths derived may not be the shortest paths, but they are at least *locally minimal*.

As a final example, let us attempt to count the number of paths between two nodes comprised of disjoint paths.

Task 4. Write a proposition which counts the number of disjoint paths from a to b so that $\mathsf{connected}(a, b, n)$ is derivable if and only if there are at least n completely disjoint paths from a to b. Assume that at the beginning of the derivation in addition to all of the propositions representing a graph that you are supplied with start.

$$\frac{\mathsf{start}}{\mathsf{connected}(a, b, z)} \qquad \frac{\mathsf{connected}(a, b, n) \quad \mathsf{path}(a, b)}{\mathsf{connected}(a, b, s(n))}$$