A Supplementary Material

A.1 Edge case handling: overlapping arguments

Occasionally, a word may be part of multiple arguments of the same connective (e.g., both the Cause and the Effect). For example, in *This equipment is newer and thus safer,* the Cause and Effect of *thus* would respectively be annotated as *this equipment is newer* and *this equipment is safer.* When processing such a shared word, the DeepCx algorithm issues an arc transition that includes both argument names (e.g., LEFT-ARC$_{\text{Cause,Means}}$). From the tagger's standpoint, this is an entirely separate transition from, say, LEFT-ARC$_{\text{Cause}}$ or LEFT-ARC$_{\text{Means}}$.

When executing an action with multiple argument types, a separate arc is added to A for each argument type—i.e., the word is added to both argument spans.

A.2 Parser details

The LSTM parser assumes its input has already been split into sentences and POS-tagged. These preprocessing steps are performed using Stanford CoreNLP (Manning et al., 2014).

A.3 Constraints on transition ordering

As mentioned in §4, several transitions have constraints on their ordering to ensure semantic well-formedness. The following constraints apply:

- A CONN-FRAG may not immediately follow a SPLIT or another CONN-FRAG.
- A SPLIT may not immediately follow a CONN-FRAG or another SPLIT.
- A SPLIT is permitted only if the connective currently under construction has at least one fragment—i.e., it contains at least two words.
- NO-CONN is forbidden if s is true, i.e., if a has been determined to be a connective anchor.

These are enforced at each timestep, both at training and test time, by eliminating violating transitions from the tagger's set of available next actions.

A.4 Neural network details and training parameters

Each LSTM is initialized with its own default item whose values are trained parameters.

We followed the training parameters of Dyer et al. (2015): we used gradient descent for parameter optimization, with an initial learning rate of $\eta_0 = 0.1$ and updates of $\eta_t = \eta_0/(1 + 0.8t)$ after each epoch t; we clipped the ℓ_2 norm of the gradients to 5; and we applied an ℓ_2 penalty of 10^{-6} to all weights. We also used Glorot initialization (Glorot and Bengio, 2010) for all parameters. Each fold took about 40 minutes to train on a single core of a 3.10-GHz processor.