An Alternative Cadence Semantics

Mark Wilding

1 The old semantics

Mark Steedman's rules to produce categories to recognise cadences work by combining the chords of the (potentially unboundedly extended) cadence itself into a category of the form (Y^7/I^7) and applying a type-raising-style rule to the target of the cadence:

$$X \to (I_X \backslash I_X) \backslash (Y^7/I_X^7)$$

The version of this given with semantics attached is:

$$X \ : \ origin \to (I_X \backslash I_X) \backslash (Y^7/I_X^7) \ : \ \lambda cadence.\lambda origin.origin + cadence(origin)$$

This includes an implicit condition of the second combination of the resulting category with another by the function application rule: namely, that, where the $(I_X \backslash I_X)$ category is the functor, the argument's semantics must be *origin*. That is to say that the semantics of the origin of cadence must be equivalent to that of the target.

This condition is overly restrictive. Whilst simple derivations are possible, no sequence can be recognised as containing consecutive cadences, since the origin of the second cadence from the end would have the semantics of one or more cadences, but it would be required to have the semantics of the target (usually just I). The example Mark gives is such a cadence and the condition is not met in this derivation.

2 An alternative

An alternative way to build the semantics of a cadence would be to forget the condition that makes the origin semantically equal to the target, since this seems fundamentally flawed. We can replace the cadence-raising rule with the following, which still requires that the origin has the same category as the target, but separates their semantics.

$$X \ : \ target \to (I_X \backslash I_X) \backslash (Y^7/I_X^7) \ : \ \lambda cadence.\lambda origin.origin + cadence(target)$$

By using this, the semantics of the target is not lost during cadence-raising, but instead is included in the semantics of the raised category.

This means that Mark's example would be given the following semantics:

$$I \ : \ (I + leftonto(I)) + leftonto(leftonto(leftonto(leftonto(leftonto(I)))))$$

This, as it happens, is exactly what Mark mistakenly wrote as the semantics for this sequence.