Logic and Language. Quiz Week 1

1 Encoding MCFG in $\mathrm{ILL}_{\multimap}$

1.1 Recap

Multiple Context-Free Grammars (MCFG) form a natural generalization of CFG: whereas in a CFG, non-terminals range over strings, in a MCFG, they range over *tuples* of strings. In the rules of a k-MCFG, the maximum number of elements of the tuples is k. An ordinary CFG, from this perspective, is simply a 1-MCFG.

As an example, consider the following 2-MCFG for $\{\mathsf{a}^n\mathsf{b}^n\mathsf{c}^n\mathsf{d}^n \mid n \geq 0\}$. We write the rules in a clausal form reminiscent of logic programming. Non-terminals are predicate symbols; their arity is the dimension of the tuple they range over. For this grammar we have $S/1$ and $A/2$.

$$
\begin{array}{l}
S(xy) \leftarrow A(x, y).\\
A(\mathsf{a}\, x\, \mathsf{b}, \mathsf{c}\, y\, \mathsf{d}) \leftarrow A(x, y).\\
A(\epsilon, \epsilon).
\end{array}
$$

We have seen how to encode such a grammar by means of a compositional translation from an Abstract Syntax source to the string language generated. The atomic types of the source signature Σ_0 are the non-terminals: $\mathcal{A}_0 = \{S, A\}$. For each rule of the grammar, there is a souce constant, with a linear implicative type read off from the rewriting rule:

$$
\begin{array}{lcl}
c_0 & :: & A \multimap S\\
c_1 & :: & A \multimap A\\
c_2 & :: & A
\end{array}
$$

The target signature Σ_1 has a single atomic type $\mathcal{A}_1 = \{*\}$. Strings are modelled as functions of type $* \multimap *$, which we abbreviate as σ. The constants of the target signature are the terminal symbols a,b,c,d of type σ.

To model an n-tuple with elements of type $A_1,\ldots,A_n$ in $\mathrm{ILL}_{\multimap}$, we can use a higher-order function of type $(A_1 \multimap \cdots \multimap A_n \multimap B) \multimap B$. Concretely, for the string tuples of our grammar, this means $(\sigma \multimap \sigma \multimap \sigma) \multimap \sigma$, which we abbreviate as $\sigma^{(2)}$.

We are ready now to specify the interpretation. The function η translates the atomic types of Σ_0: $\eta(S) = \sigma$, $\eta(A) = \sigma^{(2)}$. The function θ translates the source constants. We write (infix) $+$ as an abbreviation for function composition (=string concatenation) and ε for $\lambda i.i$ (identity function, encoding the empty string).

$$
\begin{array}{lcllcl}
\theta(c_0) & = & \lambda q.(q\ \lambda x\lambda y.(x + y)) & :: & \sigma^{(2)} \multimap \sigma\\
\theta(c_1) & = & \lambda q\lambda f.(q\ \lambda x\lambda y.(f\ (\mathsf{a} + x + \mathsf{b})\ (\mathsf{c} + y + \mathsf{d}))) & :: & \sigma^{(2)} \multimap \sigma^{(2)}\\
\theta(c_2) & = & \lambda f.(f\ \varepsilon\ \varepsilon) & :: & \sigma^{(2)}
\end{array}
$$

Below is a sample derivation for the abstract syntax term $(c_0\ (c_1\ (c_1\ c_2)))$, and for its translation, which produces the (lambda term encoding the) string $\mathsf{aabbccdd}$ after β-reduction. Computation = proof reduction!

$$
\dfrac{\dfrac{c_0}{A \multimap S} \quad \dfrac{\dfrac{c_1}{A \multimap A} \quad \dfrac{\dfrac{c_1}{A \multimap A} \quad \dfrac{c_2}{A}}{c_1\ c_2 : A}[\multimap E]}{c_1\ (c_1\ c_2) : A}[\multimap E]}{c_0\ (c_1\ (c_1\ c_2)) : S}[\multimap]
$$

$$
\widehat{\theta}(c_0\ (c_1\ (c_1\ c_2))) = \lambda i.(\mathsf{a}\ (\mathsf{a}\ (\mathsf{b}\ (\mathsf{b}\ (\mathsf{c}\ (\mathsf{c}\ (\mathsf{d}\ (\mathsf{d}\ i))))))))
$$

The complexity fingerprint of this encoding is (2,4): the maximal order of the source types is 2 (no nested implications); the maximal order of the translation of atomic source types is 4 (A is mapped to $(\sigma \multimap \sigma \multimap \sigma) \multimap \sigma$, where σ itself is of order 2 (σ abbreviates $* \multimap *$).

1.2

Consider the languages below (with $n, m > 0$, so these languages do not contain the empty string):

L_1: $\{w^2 \mid w \in \{\mathsf{a}, \mathsf{b}\}^+\}$, i.e. the copy language for non-empty words over alphabet $\{\mathsf{a}, \mathsf{b}\}$
L_2: $\{\mathsf{a}^n\mathsf{b}^n\mathsf{c}^n \mid n > 0\}$
L_3: $\{\mathsf{a}^n\mathsf{b}^m\mathsf{c}^n\mathsf{d}^m \mid n, m > 0\}$

Assignment Work out the construction of §1.1 for these grammars:

1. write a 2-MCFG for L_1–L_3
2. specify the corresponding ILL$_{\multimap}$ source and target signatures, and the translations η (source atoms to target types) and θ (source constants to target terms)
3. derive a sample string for each of the languages, i.e. give a term of the abstract source language, and show how it produces the target string under the θ translation (and β-reduction):
   - L_1: baabaa
   - L_2: aabbcc
   - L_3: abbcdd

Solutions For L_3, we have $\eta(B) = \sigma^{(2)}$; for the other non-terminals, see §1.1.

L_1:

GRAMMAR	Σ_0 (SOURCE)	$\theta(c_i)$
$S(xy) \leftarrow A(x, y).$	$c_0 :: A \multimap S$	$\lambda q.(q\ \lambda x \lambda y.(x + y))$
$A(\mathsf{a}\,x, \mathsf{a}\,y) \leftarrow A(x, y).$	$c_1 :: A \multimap A$	$\lambda q \lambda f.(q\ \lambda x \lambda y.(f\ (\mathsf{a} + x)\ (\mathsf{a} + y)))$
$A(\mathsf{b}\,x, \mathsf{b}\,y) \leftarrow A(x, y).$	$c_2 :: A \multimap A$	$\lambda q \lambda f.(q\ \lambda x \lambda y.(f\ (\mathsf{b} + x)\ (\mathsf{b} + y)))$
$A(\mathsf{a}, \mathsf{a}).$	$c_3 :: A$	$\lambda f.(f\ \mathsf{a}\ \mathsf{a})$
$A(\mathsf{b}, \mathsf{b}).$	$c_4 :: A$	$\lambda f.(f\ \mathsf{b}\ \mathsf{b})$

L_2:

GRAMMAR	Σ_0 (SOURCE)	$\theta(c_i)$
$S(xy) \leftarrow A(x, y).$	$c_0 :: A \multimap S$	$\lambda q.(q\ \lambda x \lambda y.(x + y))$
$A(\mathsf{a}\,x\,\mathsf{b}, \mathsf{c}\,y) \leftarrow A(x, y).$	$c_1 :: A \multimap A$	$\lambda q \lambda f.(q\ \lambda x \lambda y.(f\ (\mathsf{a} + x + \mathsf{b})\ (\mathsf{c} + y)))$
$A(\mathsf{ab}, \mathsf{c}).$	$c_2 :: A$	$\lambda f.(f\ (\mathsf{a} + \mathsf{b})\ \mathsf{c})$

L_3:

GRAMMAR	Σ_0 (SOURCE)	$\theta(c_i)$
$S(xzyw) \leftarrow A(x, y), B(z, w).$	$c_0 :: A \multimap B \multimap S$	$\lambda q \lambda q'.(q\ \lambda x \lambda y.(q'\ \lambda z \lambda w.(x + z + y + w)))$
$A(\mathsf{a}\,x, \mathsf{c}\,y) \leftarrow A(x, y).$	$c_1 :: A \multimap A$	$\lambda q \lambda f.(q\ \lambda x \lambda y.(f\ (\mathsf{a} + x)\ (\mathsf{c} + y)))$
$B(\mathsf{b}\,x, \mathsf{d}\,y) \leftarrow B(x, y).$	$c_2 :: B \multimap B$	$\lambda q \lambda f.(q\ \lambda x \lambda y.(f\ (\mathsf{b} + x)\ (\mathsf{d} + y)))$
$A(\mathsf{a}, \mathsf{c}).$	$c_3 :: A$	$\lambda f.(f\ \mathsf{a}\ \mathsf{c})$
$B(\mathsf{b}, \mathsf{d}).$	$c_4 :: B$	$\lambda f.(f\ \mathsf{b}\ \mathsf{d})$

(For L_3 you can also do with just A, 'growing' a,c at the front, and b,d at the back.)

Abstract syntax terms for the sample strings:

- (L_1) baabaa: $c_0\ (c_2\ (c_1\ c_3))$
- (L_2) aabbcc: $c_0\ (c_1\ c_2)$
- (L_3) abbcdd: $c_0\ c_3\ (c_2\ c_4)$

Remark The languages of this exercise are actually within the reach of a *subclass* of 2-MCFG: the *well-nested* 2-MCFG. The complexity type for that subclass is (2,3), which means you could give a simpler construction where the maximal order of the translation of atomic source types is 3 rather than 4. Concretely, the construction involves non-terminals interpreted as $\sigma \multimap \sigma$, functions from strings to strings. See (de Groote 2002, §6) on modelling Tree Adjoining Grammars.

1.3 Challenge

[This is not a hand-in exercise. If you can solve it by Dec 5, there will be a present for you!]

Let D^n be the language over an n-symbol alphabet, lexicographically ordered $a_1 < \cdots < a_n$, where words satisfy the following conditions:

1. each word contains an equal number of the n alphabet symbols
2. for every prefix p of a word, the number of a_i in $p \geq$ the number of a_{i+1} $(1 \leq i \leq n-1)$

D^n generalizes the familiar language of balanced brackets, in which case you have an alphabet of size 2, say $\{\mathsf{a}, \mathsf{b}\}$, with 'opening bracket' a preceding 'closing bracket' b in the lexicographic ordering.

The conjecture (Makoto Kanazawa, p.c.) is that for $n \geq 2$, D^n is the language of a non-wellnested $(n-1)$-MCFG.

Give a 2-MCFG for D^3, i.e. words over a 3-letter alphabet $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ (with the usual lexicographic order) satisfying conditions (1) and (2) above. Give the ACG encoding of your MCFG for D^3.

Reference M. Moortgat (2014), A note on multidimensional Dyck languages.

□