Introduction to Formal Semantics
Exam on 18.07.2022
Help Sheet

Composition Rule 1: Function Application

Let γ be a syntax tree whose sub-trees are α and β where:
- $\alpha \rightsquigarrow \alpha'$ where α' has type $\langle \sigma, \tau \rangle$
- $\beta \rightsquigarrow \beta'$ where β' has type $\langle \sigma \rangle$

then $$\gamma \rightsquigarrow \alpha'(\beta')$$

Composition Rule 2: Non-branching Nodes

If β is a tree whose only daughter is α, where $\alpha \rightsquigarrow \alpha'$ then $\beta \rightsquigarrow \alpha'$

Composition Rule 3: Predicate Modification

If:
- γ is a tree whose only two subtrees are α and β
- $\alpha \rightsquigarrow \alpha'$
- $\beta \rightsquigarrow \beta'$
- α' and β' are of type $\langle e, t \rangle$

Then:
$\gamma \rightsquigarrow \lambda u.[\alpha'(u) \wedge \beta'(u)]$
where u is a variable of type e that does not occur free in α' or β'.

Composition Rule 4: Pronouns and Trace Rule

If α is an indexed trace or pronoun, $\alpha_i \rightsquigarrow v_i$

Composition Rule 5: Predicate Abstraction

If:
- γ is a tree whose only two subtrees are α_i and β
- $\beta \rightsquigarrow \beta'$
- β' is an expression of type t

Then $\gamma \rightsquigarrow \lambda v_i.\beta'$

Type-Shifting Rule 1: Predicate-to-modifier shift

If $\alpha \rightsquigarrow \alpha'$, where α' is of type $\langle e, t\rangle$,
then $\alpha \rightsquigarrow \lambda P.[\alpha'(x) \wedge P(x)]$ (as long as P and x are not free in α; in that case, use different variables of the same type).

Type Shifting Rule 2: Object Raising (RAISE-O)

If an English expression α is translated into a logical expression α' of type $\langle e, \langle \alpha, t\rangle\rangle$ for any type α, then α also has a translation of type $\langle\langle\langle e, t\rangle, t\rangle, \langle \alpha, t\rangle\rangle$ of the following form:

$$\lambda Q_{\langle\langle e,t\rangle,t\rangle} \lambda x_{\alpha}.Q(\lambda y.\alpha'(y)(x))$$

(unless Q, y or x occurs in α'; in that case, use different variables).

Type Shifting Rule 3: Subject Raising (RAISE-O)

If an English expression α is translated into a logical expression α' of type $\langle \alpha, \langle e, t\rangle\rangle$ for any type α, then α also has a translation of type $\langle \alpha \langle\langle\langle e, t\rangle, t\rangle, t\rangle\rangle$ of the following form:

$$\lambda y_{\alpha} \lambda Q_{\langle e,t\rangle,t\rangle}.Q(\lambda x_e.\alpha'(y)(x))$$

(unless Q, y or x occurs in α'; in that case, use different variables).

Type-Shifting Rule 5: Existential Closure

if $\alpha \rightsquigarrow \alpha'$, where α' is of a category $\langle v, t\rangle$, then:

$$\alpha \rightsquigarrow \exists e.\alpha'(e)$$

as well (as long as) e does not occur in α'; in that case, use a different variable of the same type

Syntax Rule: Definedeness Conditions

If ϕ is an expression of type t , then $\partial(\phi)$ is an expression of type t

Semantic Rule: Defineteness Conditions

If ϕ is an expression of type t , then:

$$[\![\partial(\phi)]\!]^{M,g} = \begin{cases} 1 \; if \; [\![\phi]\!]^{M,g} = 1 \\ \#_e otherwise \end{cases}$$

Semantic Rule: Existence Predicate

$[\![Exists(\alpha)]\!]^{M,g} = 1$ if $[\![\alpha]\!]^{M,g} \neq \#_e$ and 0 otherwise

Type-Shifting Rule 6: Quantifier Closure

if $\alpha \rightsquigarrow \alpha'$, where α' is of a category $\langle\langle v, t\rangle, t\rangle$, then:
$\alpha \rightsquigarrow \alpha'(\lambda e.true)$
as well.