Language B

The language B consists of statements of the forms described and explained below. These statements are composed of keywords and of the names of the individuals, types, and relations of the respective mini world. The used keywords are „HasType“, „SubTypeOf“, „not“, „some“, and „only“.

Statements

Every statement can either be true or false. Every statement of the language B has the form of one of the four schemes described here. Note that types can be complex (see the next section).

Positive simple statements	
scheme:	*Individual1* *Relation* *Individual2*
example:	John sees Mary
explanation:	A positive simple statement consists of two individuals and one relation and states that the first individual has the given relation to the second individual. The example above states that John has a „sees”-relation to Mary.

Negative simple statements	
scheme:	*Individual1* **not** *Relation* *Individual2*
example:	Mary **not** helps Bill
explanation:	A negative simple statement consists of two individuals and one relation where the relation is preceded by the keyword „not“. Such a statement states that the first individual does not have the given relation to the second individual. The example above states that Mary does not have a „helps“-relation to Bill.

HasType-statements	
scheme:	*Individual* ***HasType*** *Type*
example:	John **HasType** man
explanation:	A HasType-statement requires an individual and a type and it states that the given individual belongs to the given type. The example above states that John is a man.

SubTypeOf-statements	
scheme:	*Type1* **SubTypeOf** *Type2*
example:	golfer **SubTypeOf** man
explanation:	A SubTypeOf-statement requires two types and states that every individual that belongs to the first type also belongs to the second type (but not necessarily the other way round). The example above states that every individual that is a golfer is also a man.

Type Operators

Every type (simple or complex) stands for a certain group of individuals. „woman“ and „golfer“ are examples of simple types. Apart from simple types, there are complex types that are composed by the type operators described here. „sees **only** golfer“ is an example of a complex type. Note that such complex types can be nested. In this case, parentheses are used to clarify the structure, for example „**not** (loves **some** woman)“.

not-operator	
scheme:	**not** *Type*
example:	**not** golfer
explanation:	The not-operator requires just one type. The resulting complex type stands for all individuals that do not belong to the given type. The example above stands for all individuals that are not golfers.

some-operator	
scheme:	*Relation* **some** *Type*
example:	loves **some** woman
explanation:	The some-operator requires a relation and a type. The resulting complex type stands for all individuals that have the given relation to at least one individual of the given type. The example above stands for all individuals that have a „loves“-relation to at least one woman.

only-operator	
scheme:	*Relation* **only** *Type*
example:	helps **only** woman
explanation:	The only-operator requires a relation and a type. The resulting complex type stands for all individuals that either have no corresponding relation to another individual at all, or if they do then only to individuals of the given type. The example above stands for all individuals that have „helps“-relations (if present at all) only to women. Thus, the example includes all individuals except those that have a „helps“-relation to an individual that is not a woman.