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 and types of the respective mini world. The used keywords are „HasType“, „SubTypeOf“, „DisjointWith“, „EquivalentTo“, „not“, „and“, and „or“.

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).

HasType-statements	
scheme:	*Individual* ***HasType*** *Type*
example:	John **HasType** golfer
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 golfer.

DisjointWith-statements	
scheme:	*Type1* **DisjointWith** *Type2*
example:	woman **DisjointWith** golfer
explanation:	A DisjointWith-statement requires two types and states that no individual belongs to the first type and at the same time to the second type. The example above states that no individual is a woman and is a golfer.

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.

EquivalentTo-statements	
scheme:	*Type1* **EquivalentTo** *Type2*
example:	golfer **EquivalentTo** man
explanation:	An EquivalentTo-statement requires two types and states that every individual that belongs to the first type also belongs to the second type and vice versa. The example above states that every individual that is a golfer is also a man and vice versa.

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. „woman **or** 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** (golfer **and** man)“.

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.

or-operator	
scheme:	*Type1* **or** *Type2*
example:	woman **or** golfer
explanation:	The or-operator requires two types. The resulting complex type stands for all individuals that belong to the first type or to the second type or to both. The example above stands for all individuals that are women or golfers or both.

and-operator	
scheme:	*Type1* **and** *Type2*
example:	golfer **and** man
explanation:	The and-operator requires two types. The resulting complex type stands for all individuals that belong to the first type and at the same time to the second type. The example above stands for all individuals that are golfers and men.