This document describes some things about OpenCyc in a conventional notation. A bit about the Z notation used:

:	specifies the type of a variable.
•	to bind propositions to declared variables.
|	to specify extra constraints to declared variables.

The language CycL

todo: describe Alfabet, symbols, variables, constants, terms, formulas and predicates.

Microtheory

A Microtheory Σ is a set of axioms.

Z: $\varphi \in \Sigma$
OpenCyc: `(#$ist-Asserted` Σ φ`)`

Z: $\Sigma \vdash \varphi$
OpenCyc: `(#$ist` Σ φ`)`

Collections and sets

Sets are defined extensionally, collections are defined intensionally.

Z: $x : A$
OpenCyc: `(#$isa` x A`)`

Z: $y \in B$
OpenCyc: `(#$elementOf` y B`)`

Rule Macro Predicates

A `#$RuleMacroPredicate` is a predicate that can be used to write a long implication as a short ground atomic formula.[1]. A list of rule macro predicates is shown by asking

```
(#$and
   (#$isa ?PRED #$RuleMacroPredicate)
   (#$expansion ?PRED ?TEMPLATE))
```

$$genls(A, B) \equiv \forall x : A \bullet x : B \tag{1}$$

$$genlPreds(P, Q) \equiv \forall x \bullet P(x) \Rightarrow Q(x) \tag{2}$$

$$genlMt(M, N) \equiv N \vdash \varphi \Rightarrow M \vdash \varphi \tag{3}$$

$$disjointWith(A, B) \equiv \forall x \bullet \neg(x : A \wedge x : B) \tag{4}$$

[1] ground = no variables, atomic is a formula of the form (pred ARG_1 ... ARG_k)

$$argIsa(P, C, n) \equiv \forall x_1, .., x_m \mid 1 \leq n \leq m \bullet \ P(x_1, .., x_m) \Rightarrow x_n : C \tag{5}$$

$$argGenl(P, C, n) \equiv \forall x_1, .., x_m \mid 1 \leq n \leq m \tag{6}$$
$$\bullet \ P(x_1, .., x_m) \Rightarrow x_n \subseteq C \tag{7}$$

$$interArgIsa1\text{-}2(P, A, B) \equiv \forall x : A \bullet P(x, y) \Rightarrow y \in B \tag{8}$$

$$relationAllExists(P, A, B) \equiv \forall x : A \bullet \exists y : B \bullet P(x, y) \tag{9}$$

$$relationAllInstance(P, A, y) \equiv \forall x : A \bullet P(x, y) \tag{10}$$

$$typeGenls(T, B) \equiv \forall \ C : T \bullet C \subseteq B \tag{11}$$

$$transitiveViaArg(P, Q, n) \equiv \tag{12}$$
$$\forall x_1, .., x_m, y, z \mid 1 \leq n \leq m \wedge$$
$$y, z \text{ zijn } n^{de} \text{ argument van } P \ \bullet$$
$$P(x_1, .., x_{n-1}, y, x_{n+1}, ..x_m) \wedge Q(y, z) \Rightarrow P(x_1, .., x_{n-1}, z, x_{n+1}, ..x_m)$$

$$functionalInArgs(P, n) \equiv \forall x_1, .., x_m, y \mid 1 \leq n \leq m \bullet \tag{13}$$
$$\mid \{ \ y \mid P(x_1, .., x_{n-1}, y, x_{n+1}, ..x_m) \ \} \mid \leq 1$$

$$genlInverse(P, Q) \equiv \forall x, y : Thing \bullet P(x, y) \Rightarrow Q(y, x) \tag{14}$$

$$negationPreds(P, Q) \equiv \forall x : Thing \bullet \neg(P(x) \wedge Q(x)) \tag{15}$$

KB dependent relations

$$\vdash \varphi \Rightarrow knownSentence(\varphi) \tag{16}$$

$$\nvdash \varphi \Rightarrow unknownSentence(\varphi) \tag{17}$$

Negation as failure

$$completeCollectionExtent(A) \equiv \tag{18}$$
$$\forall x : A; M : Microtheory \ \bullet \ \neg(\text{"}x \in A\text{"} \in M) \Rightarrow \neg(x \in A)$$

$$completeExtentKnown(P) \equiv \tag{19}$$
$$\forall x; M : Microtheory \bullet \neg(\text{"}P(x)\text{"} \in M) \Rightarrow \neg P(x)$$

$$minimizeExtent(P) \equiv \forall x \bullet \nvdash (P(x) \Rightarrow \neg P(x) \tag{20}$$

Monotonic vs default

A formula φ is called monotonic, if for any axiom system (microtheory) N and set of axioms Δ holds:

$$N \models \varphi \Rightarrow N \cup \Delta \models \varphi \tag{21}$$