Usage

Im Interpreter.CLI Ordner

```
dotnet run
```

oder die .exe ausführen

Verwendung mit Command Line Arguments

Die zu ladende Datei mit der Endung .lp kann mit dem Command '-f' bzw. '--file' und dem Pfad der Datei gestartet werden:

```
dotnet run --file "/path/to/your/example_file.lp"
```

Zusätzlich kann eine Query Abfrage auf die geladene Datei hinzugefügt werden:

```
dotnet run --file "/path/to/your/example_file.lp" --query "your query here"
```

Verwendung der CLI

Durch einfaches Starten der Executable wird die implementierte CLI gestartet. Diese unterstützt folgende Befehle:

- :l "filepath" | :load "filepath" – Laden einer Datei mit der Endung .lp

```
:load "/path/to/your/example_file.lp"
```

- :ex "filepath" | :explain "filepath" - Erklären einer Datei mit der Endung .lp

```
:explain "/path/to/your/example_file.lp"
```

- :r | :reload – Erneutes Laden der CLI

- :e | :exit – Schließen der CLI
- :i | :info – Anzeigen von Informationen zur geladenen Datei
- :q "query" | :query "query" – Ausführen einer gegebenen Abfrage auf eine bereits geladene Datei

```
:query "your query here"
```

Zusätzliche Informationen (& Abweichungen)

Is Operator

Berechnet einen Ausdruck und speichert das Ergebnis.

> R is X + 2.

Range Operator

Definiert einen Bereich von Werten.

> num(1..10) erzeugt die Werte num(1) bis num(10).

Choices

Erlaubt, dass die Literale in den Klammern entweder true oder false sein können.

> {a; b; c}.

ODER im Körper einer Regel

Ein Semikolon ; im Körper einer Regel stellt ein logisches ODER dar.

> ist_informatiker(X) :- mensch(X); alien(X), hatBeruf(X, informatiker).

Kopflose Regeln

Definiert etwas was unter keinen Umständen in einem set gelten darf.

```
:- ist_informatiker(nina).
```

Verschachtelungen

Erlaubt das verschachteln von Fakten.

```
ist_informatiker(frau(mensch(X))).
```

Zahlen

Sowohl positive als auch negative Zahlen.

```
node(2.) node(-30). num(-3..7).
```

Comparison

Vergleichsoperatoren wie ==, !=, <, >, <=, >= werden verwendet um Werte zu vergleichen.

```
X == Y. X <= Y
```

Unifikation

Setzt zwei Terme gleich und prüft, auf die Struktur dieser.

```
X = Y.
```

Classical Negation

Stellt die Negation einer Tatsache dar.

```
-p(X) bedeutet, dass p(X) falsch ist.
```

Negation as Failure

Stellt die Annahme dar, dass etwas nicht wahr ist, weil es nicht bewiesen werden kann.

> not p(X) bedeutet, dass p(X) nicht bewiesen werden kann.

Anonyme Variable

Werden verwendet, wenn der konkrete Wert einer Variablen nicht relevant ist.

> square(X,_).

Requirements

> .NET 8.0 ! Important !