The package `fodot`*

Christian Fleiner
christian.fleiner@kuleuven.be

August 29, 2025

Abstract

The package `fodot` provides helpful commands to work with the FO(·) language in LaTeXincluding syntax highlighting in listings.

1 Contributing

Contributions are always welcome. The project is hosted at `https://gitlab.com/EAVISE/CFL/fodot-latex`.

2 Description

In the following, the capabilities of the package are described and illustrated. The FO(·) language itself is not introduced. FO(·)[1] is the technical implementation of FO[·][2] used in the reasoning engine IDP-Z3[3]. Please refer to the official documentation to learn more.

2.1 Commands

The following commands are currently supported:

1. Type `\fodot` for: FO(·). *(partial implementation of FO[·])*
2. Type `\fodott` for: FO[·]. *(formal knowledge representation language)*

2.2 FODOT Syntax Highlighting

There are two styles to highlight FO(·) code in listings. FO(·) code must be always copied in ASCII format. `fodot` replaces operators with their UTF-8 symbol. `fodotASCII` keeps the ASCII representations.

There are a few bugs that require workarounds:

*This document corresponds to the version 0.0.1 of `fodot`, at the date of 2025/08/29.

[1] `https://docs.idp-z3.be/en/stable/introduction.html`

[2] `https://fo-dot.readthedocs.io/en/latest/FO-dot.html`

[3] `https://idp-z3.be/`

- `*` is somehow not recognized. Replace `*` with `TIMES` in listing.
- `=<` is highlighted but not replaced (while `<=` would work). Instead, replace `=<` with `=<` in listing (not relevant for `fodotASCII`)

The syntax highlighter does not support patterns to highlight URLs.
See the following templates for each style:

```
\begin{lstlisting}[style=fodot]
...
\end{lstlisting}

\begin{lstlisting}[style=fodotASCII]
...
\end{lstlisting}
```

2.2.1 Examples: Fodot Style

Code snippet A

```
vocabulary {
    // this is a single-line comment
    type Method ≜ {Nail, Glue, Screw}
    type Wall ≜ {Brick, Wood, Tile}
    [this is an annotation]
    type Difficulty ≜ {1..3}

    /*
    this is a block comment
    */
    wall  : () → Wall
    method: () → Method
    hole  : () → 𝔹
    weight: () → ℤ
    difficulty : () → Difficulty
}
theory {
    weight() > 0.
    method() = Nail  ⇒ weight() ≤ 25.
    method() = Screw ⇒ weight() ≤ 40.
    method() = Glue  ⇒ weight() ≤ 15.

    hole()  ⇔  method() = Nail ∨ method() = Screw.
    wall() = Tile ⇒ ¬hole().

    { difficulty() = 1 ← method() = Glue.
    difficulty() = 2 ← method() = Nail.
    difficulty() = 3 ← method() = Screw.}
}
display {
```

```
31      view() = expanded.
32  }
```

Code Snippet B

```
@prefix we: <http://www.example.org/whatever#>.
vocabulary V {
    @prefix se: <http://www.example.org/somethingelse#_url>
        .
    type T
    type T ≜ {c1, c2, c3}
    type T ≜ constructed from {c1, c2(T1, f:T2)}
    type T ≜ {1,2,3} ⊆ ℤ
    type T ≜ {1..3} ⊆ ℤ
    type we::T
    type <http://www.example.org/foo#Type>

    p : () → 𝔹
    p1, p2 : T1 × T2 → 𝔹
    f: total T → T
    f: T * T → T  (domain: p, codomain: q)
    f: partial T×T → T
    f1, f2: Concept[T1→T2] → T

    [this is the intended meaning of p]
    p : () → 𝔹

    var x in T
    import W
}

theory T:V {
    (¬p1()∧p2() ∨ p3() ⇒ p4() ⇔ p5()) ⇐ p6().
    p(f1(f2())).
    f1() < f2() ≤ f3() = f4() ≥ f5() > f6().
    f() ≠ c.
    ∀x,y in T: p(x,y).
    ∀x in p, (y,z) in q: q(x,x) ∨ p(y) ∨ p(z).
    ∃x in Concept[()→𝔹]: $(x)().
    ∃x: p(x).  # if var x declared in voc
    ∃>1 x in T: p(x).

    f() in {1,2,3}.
    f() = #{xinT: p(x)}.
    f() = min{  f(x) ∨ x in T: p(x) }.
    f() = sum{{ f(x) ∨ x in T: p(x) }}.
    if p1() then p2() else p3().
    f1() = if p() then f2() else f3().

    p ≜ {1,2,3}.
    p(#2020-01-01) is enumerated.
    p(#TODAY) is not enumerated.

```

```
48      { p(1). }
49      { (co-induction)
50        ∀xinT: p1(x) ← p2(x).
51        f(1)=1.
52        ∀x: f(x)=1 ← p(x).
53        ∀x: f(x) ≜ 1 ← p(x).
54      }
55
56      [this is the intended meaning of the rule]
57      p().
58  }
59
60  structure S:V {
61      p ≜ false.
62      p ≜ {1,2,3}.
63      p ≜ {0..9, 100}.
64      p ≜ {#2021-01-01}.
65      p ≜ {(1,2), (3,4)}.
66      p ≜ {
67      1 2
68      3 4
69      }.
70
71      f ≜ 1.
72      f ≜ {→1} .
73      f ≜ {1→1, 2→2}.
74      f ≜ {(1,2)→3} else 2.
75      f ⊇ {(1,2)→3}.
76  }
77
78  display {
79      goal_symbol ≜ {`p1, `p2}.
80      hide(`p).
81      expand ≜ {`p}.
82      view() = expanded.
83      optionalPropagation().
84  }
85
86  procedure main() {
87      pretty_print(model_check    (T,S))
88      pretty_print(model_expand   (T,S))
89      pretty_print(model_propagate(T,S))
90      pretty_print(minimize(T,S, term="cost()"))
91  }
```

2.2.2 Examples: FodotASCII Style

Code Snippet A

```
vocabulary {
    // this is a single-line comment
    type Method := {Nail, Glue, Screw}
    type Wall := {Brick, Wood, Tile}
    [this is an annotation]
    type Difficulty := {1..3}

    /*
    this is a block comment
    */
    wall  : () -> Wall
    method: () -> Method
    hole : () -> Bool
    weight: () -> Int
    difficulty : () -> Difficulty
}
theory {
    weight() > 0.
    method() = Nail  => weight() =&lt; 25.
    method() = Screw => weight() =< 40.
    method() = Glue  => weight() =< 15.

    hole() <=> method() = Nail | method() = Screw.
    wall() = Tile => ~hole().

    { difficulty() = 1 <- method() = Glue.
    difficulty() = 2 <- method() = Nail.
    difficulty() = 3 <- method() = Screw.}
}
display {
    view() = expanded.
}
```

Code Snippet B

```

@prefix we: <http://www.example.org/whatever#>.
 vocabulary V {
     @prefix se: <http://www.example.org/somethingelse#_url>
         .
     type T
     type T := {c1, c2, c3}
     type T := constructed from {c1, c2(T1, f:T2)}
     type T := {1,2,3} <: Int
     type T := {1..3} <: Int
     type we::T
     type <http://www.example.org/foo#Type>

     p : () -> Bool
     p1, p2 : T1 × T2 -> Bool
     f: total T -> T
     f: T * T -> T  (domain: p, codomain: q)
     f: partial T×T -> T
     f1, f2: Concept[T1->T2] -> T

     [this is the intended meaning of p]
     p : () -> Bool

     var x in T
     import W
 }

 theory T:V {
     (~p1()&p2() | p3() => p4() <=> p5()) <= p6().
     p(f1(f2())).
     f1() < f2() =< f3() = f4() >= f5() > f6().
     f() ~= c.
     !x,y in T: p(x,y).
     !x in p, (y,z) in q: q(x,x) | p(y) | p(z).
     ?x in Concept[()->Bool]: $(x)().
     ?x: p(x).  # if var x declared in voc
     ?>1 x in T: p(x).

     f() in {1,2,3}.
     f() = #{xinT: p(x)}.
     f() = min{  f(x) | x in T: p(x) }.
     f() = sum{{ f(x) | x in T: p(x) }}.
     if p1() then p2() else p3().
     f1() = if p() then f2() else f3().

     p := {1,2,3}.
     p(#2020-01-01) is enumerated.
     p(#TODAY) is not enumerated.
```

```
48
49      { p(1). }
50      { (co-induction)
51        !xinT: p1(x) <- p2(x).
52        f(1)=1.
53        !x: f(x)=1 <- p(x).
54        !x: f(x):=1 <- p(x).
55      }
56
57      [this is the intended meaning of the rule]
58      p().
59  }
60
61  structure S:V {
62      p := false.
63      p := {1,2,3}.
64      p := {0..9, 100}.
65      p := {#2021-01-01}.
66      p := {(1,2), (3,4)}.
67      p := {
68      1 2
69      3 4
70      }.
71
72      f := 1.
73      f := {->1} .
74      f := {1->1, 2->2}.
75      f := {(1,2)->3} else 2.
76      f :> {(1,2)->3}.
77  }
78
79  display {
80      goal_symbol := {`p1, `p2}.
81      hide(`p).
82      expand := {`p}.
83      view() = expanded.
84      optionalPropagation().
85  }
86
87  procedure main() {
88      pretty_print(model_check    (T,S))
89      pretty_print(model_expand   (T,S))
90      pretty_print(model_propagate(T,S))
91      pretty_print(minimize(T,S, term="cost()"))
92  }
```