

Институт математики, механики и компьютерных наук имени И. И. Воровича

02.03.02 – Фундаментальная информатика и информационные технологии

# Разработка плагина языка **Coq** для редактора **Atom**

**Выполнил:** Стребежев. И.А.

**Научный руководитель:** к.т.н. ст. преп. Алымова Е.В.

# Постановка задачи

1. Подсветка синтаксиса
2. Снимпеты, автодополнение
3. Компиляция предложений по шагам
4. Запоминание вычисленных результатов
5. Генерация веток для сопоставление с образцом
6. Автодополнение, основанное на выводе типов

# Подсветка синтаксиса

```
01. inductive_type_decl:
02.   comment: "Inductive type declarations"
03.   match:   "(CoInductive|Inductive)\\s+([a-z][a-z0-9_\\']*)"
04.   captures:
05.     1:
06.       name: "keyword.source.coq"
07.     2:
08.       name: "entity.name.type.coq"
```

```
7  Eval compute in 1 + 1 * 2 = 2.
8  Eval compute in "asasda".
9
10 (* ssreflect *)
11 Eval compute in if: (1 + 1 * 2 == 2) then: True else: False.
12
13 Definition next (x : nat) : nat :=
14   match x with
15   | 0 => S 1 + 2
16   | _ => 2
17   end.
18
19 Lemma prime_above : True.
20 Proof.
21   by rewrite addn1 ltnS fact_gt0.
22   exists p => //; rewrite ltnNge; apply: contraL p_dv_m1 => p_le_m.
23   admit.
24 Qed.
```

# Сниппеты

Встраиваемый блок кода или контекстная подсказка.

- Список модулей
- Список тактик
- Основные операторы

**Проблема!** тактик более **200**, имеют смысл *только внутри* доказательств

**Проблема!** компилятор не предоставляет контекстных подсказок (вообще)

```
7 Eval compute in 1 + 1 * 2 = 2.
```

```
8 Eval compute in "asasda".
```

```
9
```

```
10 (* ssreflect *)
```

```
11 Eval compute in if: (1 + 1 * 2 == 2) then: True else: False.
```

```
12 defi
```

➔ **Definition** Definition ... := ... it :=

**definitions**

**definition**

**Defined**

**Definitions**

**Defensive**

```
21 by rewrite addn1 ltnS fact_gt0.
```

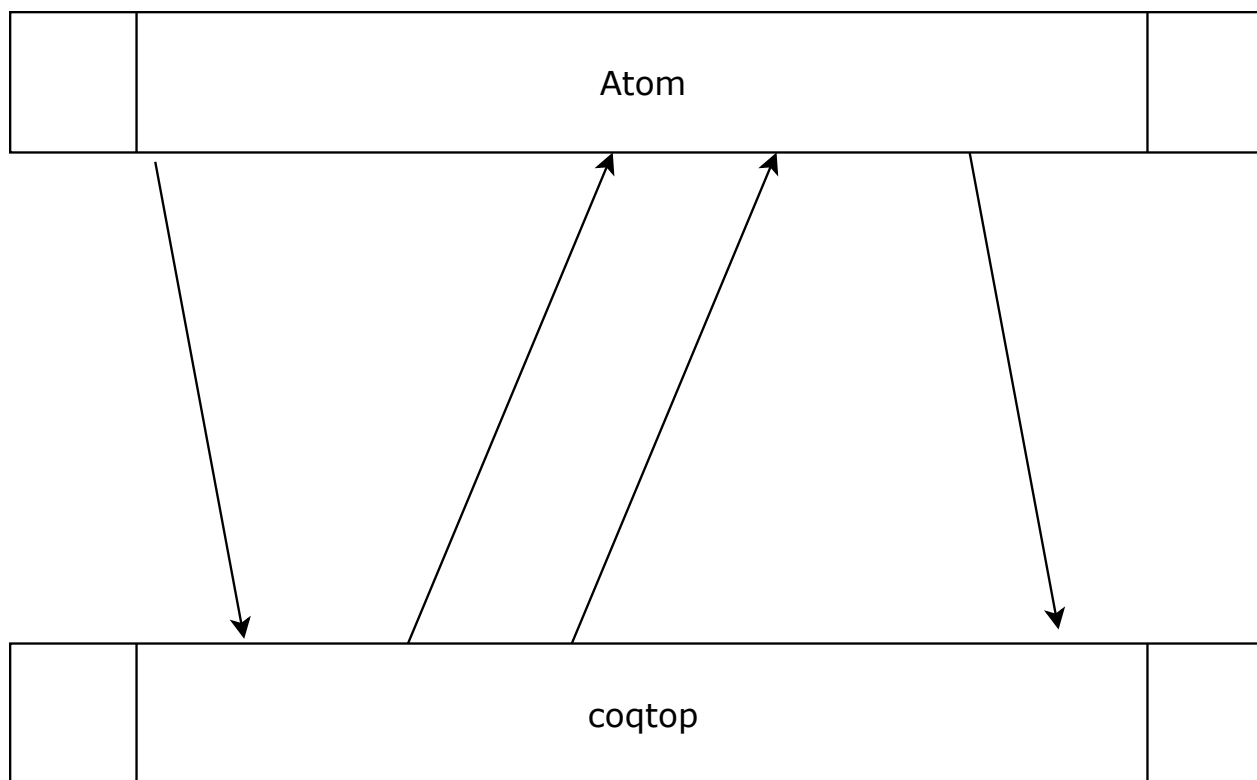
```
22 exists p => //; rewrite ltnNge; apply: contraL p_dv_m1 => p_le_m.
```

```
23 admit.
```

```
24 Qed.
```

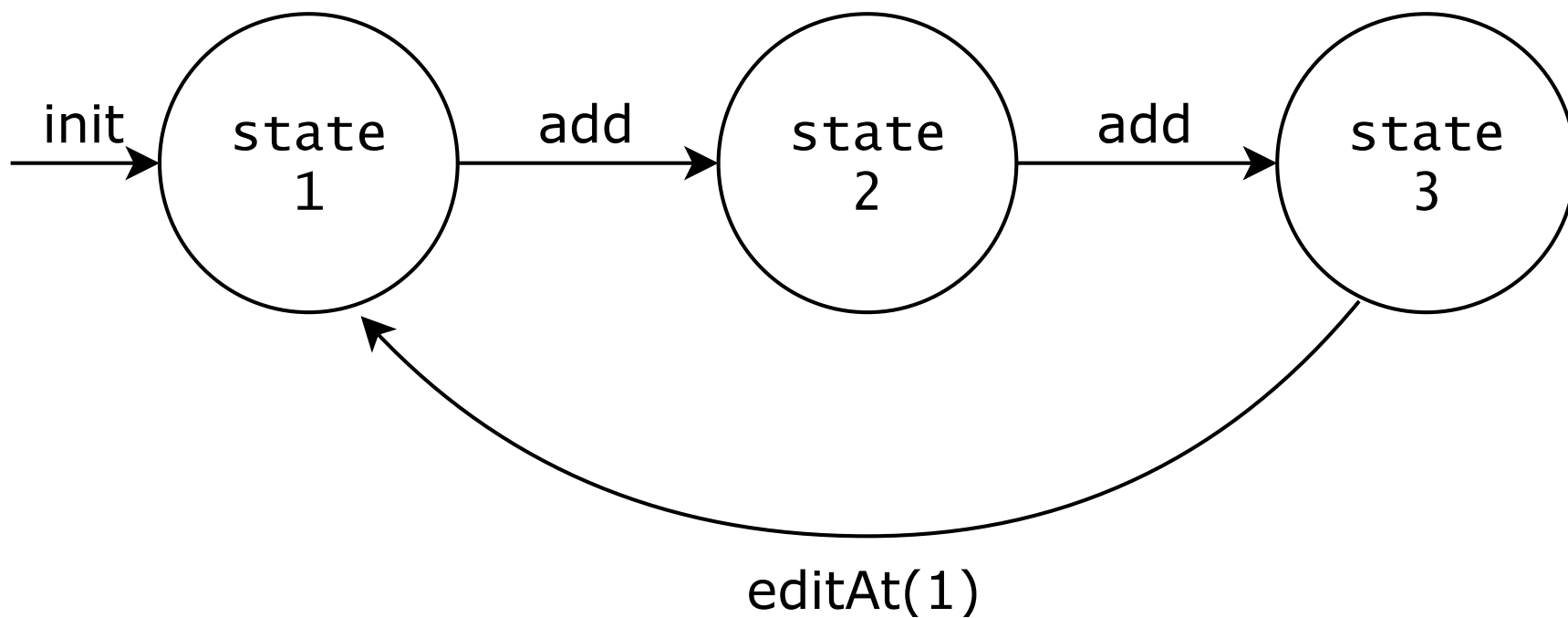
# Coqtop

`coqtop` – дочерний процесс в интерактивном режиме



# Компиляция по шагам

Каждое предложение получает свой stateld после компиляции.





```
1 From Coq Require Import Init.Prelude Unicode.Utf8.
2 From mathcomp.ssreflect Require Import ssreflect
• ssrfun ssrbool eqtype ssrnat div prime.
3 Set Implicit Arguments. Unset Strict Implicit.
4 Unset Printing Implicit Defensive.
5 Module kek.
6
7 Eval compute in 1 + 1 * 2 = 2.
8 Eval compute in "asasda".
9
10 (* ssreflect *)
11 Eval compute in (1 + 1 * 2 === 2).
12
```

# Запоминание промежуточных результатов

- Скомпилированные предложения могут вернуть результат
- Деревья вывода меняются в процессе доказательства

```
7  Eval compute in 1 + 1 * 2 = 2.      True: bool
8  Eval compute in "asasda".          "asasda": string
```

Этой функциональности нет в других редакторах

# Сопоставление с образцом

Пусть `n : nat` – переменная

Нужно знать определение типа `nat`, чтобы сделать сопоставление

```
match n with  
| 0      => "one"  
| S n'   => "two"  
end.
```

Вызывается метод компилятора для получения конструктора имён `nat`

# Результат работы

- Подсветка синтаксиса
- Снимпеты, автодополнение
- Компиляция предложений по шагам
- Запоминание вычисленных результатов
- Генерация веток для сопоставление с образцом
- Выложен в открытый доступ: <https://github.com/xamgore/language-cog>