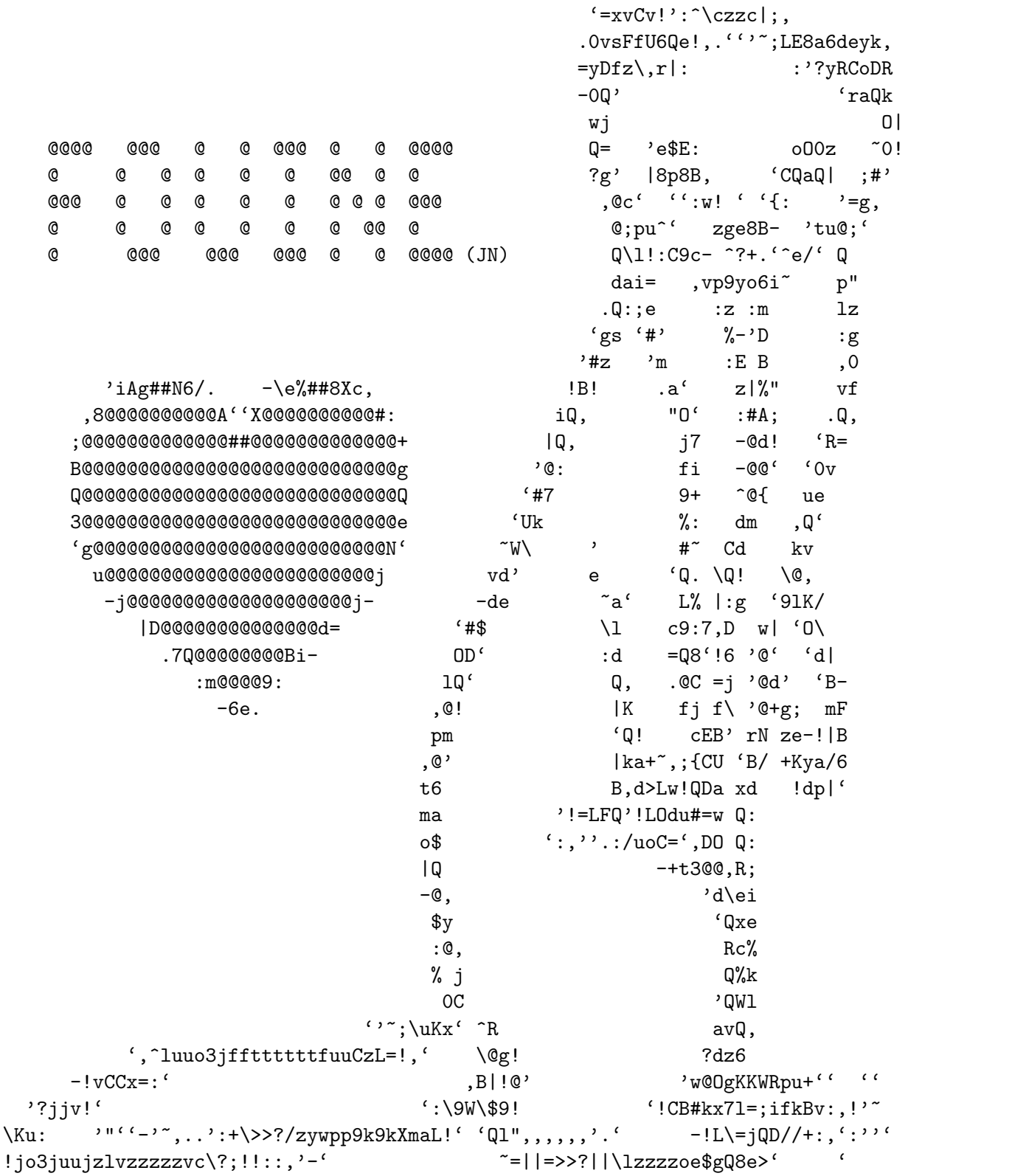
FouineJN, un langage Jovial et Naturel

or How I learned to stop worrying and love CPS

Robin Jourde & Nicolas Nardino

mai 2021


@@@@ @@@ @ @ @@@ @ @ @@@@
@ @ @ @ @ @ @@ @ @
@@@ @ @ @ @ @ @ @ @ @@@
@ @ @ @ @ @ @ @@ @
@ @@@ @@@ @@@ @ @ @@@@ (JN)

À l'origine fut la vitesse, la pure programmation fonctionnelle, le ¡¡ $\lambda x.e$ ¿¿
Puis l'évaluation décéléra, prit consistance et forme, jusqu'aux lenteurs habitables, jusqu'au `let`, jusqu'à vous.
Bienvenue à toi, lente fouine liée, poussif tresseur de fonctions récursives.

[Dam07]

Table des matières

1 Présentation

Foino (Martes foina) aŭ estas specio de marteso kiu havas brilan blugrizan ĝis blubrunan hararon, iom malpli delikatan ol tiu de la arbara marteso ; duobla makulo sur la gorĝo estas blanka. [Vik21]

`fouine` est un interpréteur de FouineJN (Fouine Joviale et Naturelle [1]), sous-ensemble du langage de programmation national, OCaml.

`fouine` est programmé principalement OCaml (1), et le parser associé est généré avec Menhir (2).

1. [Citation Needed]

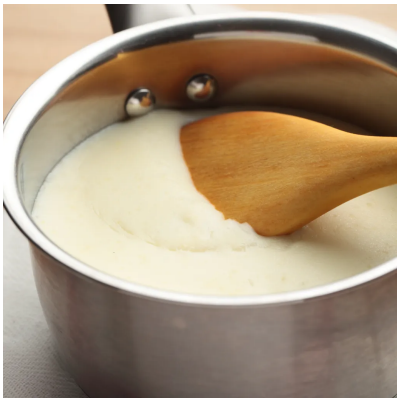


FIGURE 1 – Exemple de béchamel, dans laquelle on peut trouver `caml`

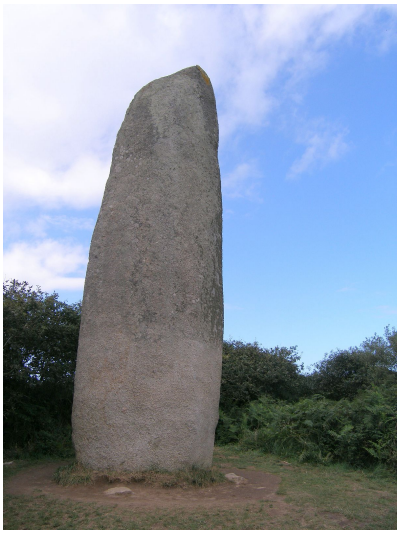


FIGURE 2 – Exemple de menhir

2 Organisation et répartition du travail

2.1 Organisation

Comme dit plus haut, `fouine` est prorammé principalement en OCAML, avec pas moins de 12 fichiers `.ml`[2] ! Nous les détaillerons plus tard. Ils sont accompagnés d'un `lexer.mll` et `parser.mly`, qui à eux deux, spécifient la grammaire de FOUINEJN.

Dans un ordre peu ou prou chronologique, les fichiers `.ml` sont :

expr.ml La base de la pyramide, c'est ici qu'on définit les expressions.

eval.ml Le cœur du travail, c'est là que la fouine travaille

2. Source : `ls -al | grep -c "\.ml$"`

main.ml L'enrobage

affichage.ml Parce que c'est pratique d'avoir un retour sur ce qui se passe

memory.ml Pouvoir se souvenir de choses est un avantage évolutif certain, pas étonnant qu'une fouine en bénéficie

stdLib.ml Des fonctions tout plein (en particulier, nos opérateurs infixes)

transformation.ml Gère le transformation CPS des programmes en entrée

reduction.ml Aide le fichier précédent dans sa quète

types.ml On type enfin notre langage ! Analogue de `expr.ml`

inference.ml L'inférence de type monomorphe

polymorphisme.ml blablabla polymorphe

unification.ml Ze algorithme of unification !

Un, deux, trois... Douze ! Le compte y est.

2.2 La répartition

Au commencement, il n'y avait rien... Enfin, il y avait le mini-projet de PROJ1 de N. (noms masqués pour préserver l'anonymité), qui a peut-être un peu aidé. Mais le vaillant R. dit : ¡¡ Que les expressions soient ¿¿ et les expressions furent. Et avec ces dernières, l'évaluation.

Mais travailler avec des expressions brutes, ce n'est pas pratique, et alors N. parsa. Et parsa. Et parsa. Et il vit que le parser était bien (*spoiler alert*, il n'était pas si bien que ça).

Quand vint le filtrage, un singulier shift/reduce conflict nous embêta des semaines durant. Il s'agissait de pouvoir imbriquer deux `match`. Jusqu'au jour de la Libération, quand le sage D.H. nous en libéra.

Entretemps, les continuations arrivèrent, et R. entreprit de métamorphoser ce que les vils utilisateurs envoyaient à notre bien-aîmée `fouine` en CPS, tandis que N. passait `eval` en CPS, afin de pouvoir traiter les exceptions. La transformation posa quelques problèmes à R., notamment pour les fonctions récursives, et il dut utiliser la ruse afin de triompher (*i.e.* des réductions).

Mais le triomphe fut court, et le typage pointa le bout de sa truffe. Alors que N. inférait monomorphiquement, R. unifiait les peuples sous sa banière, et quand ce fut fait, et que le calme retomba, N. put enfin s'atteler aux réparations tant attendues du parser, alors que R. s'adonna à son chef-d'œuvre, le *plusieurs-formes*.

3 Collaboration

3.1 git

Il était évident que du contrôle de version allait être nécessaire, ainsi qu'un dépôt distant. Nous avons donc créé un dépôt sur `github`, que l'on peut retrouver ici : `github.com/vecteurNabla/proj2`.

Grâce à notre sens de l'organisation hors du commun, notre répartition exemplaire du travail, ainsi que qu'une bonne dose de chance sûrement, il n'y eu que très peu de conflits à résoudre, et tous furent résolus aisément, dans le plus grand professionnalisme, non sans l'aide de porcelaines comme `magit` [3]

3.2 Postérité

Notre projet est extrêmement populaire, comme le montre le graphique suivant :

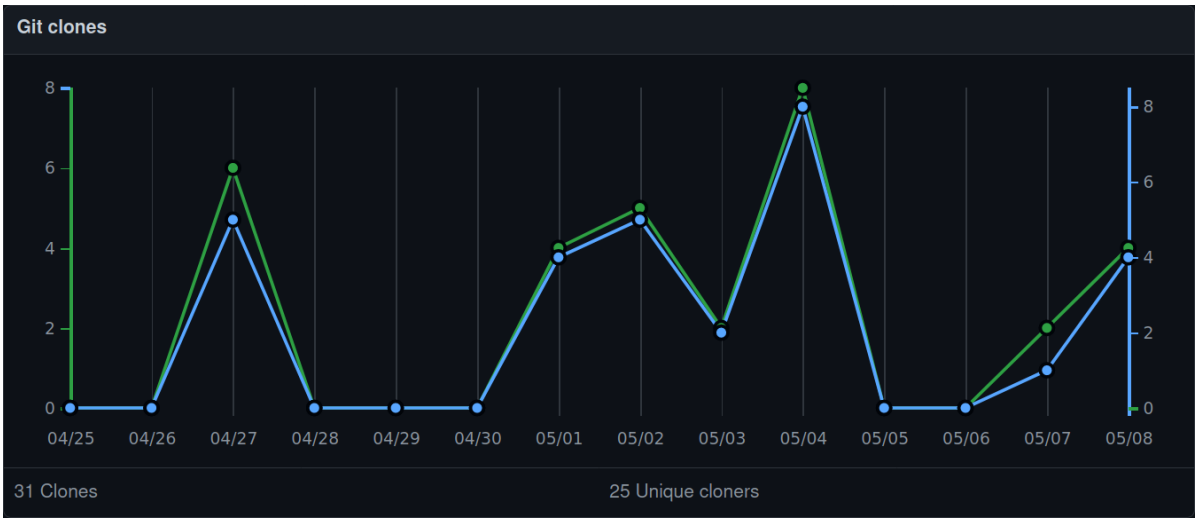


FIGURE 3 – Oh c'est joli c'est bleu

On peut observer pas moins de 31 clones en deux semaines !
[Lan66].
TCHANA2025 [NTT+21]

Références

[Dam07] Alain Damasio. *La Horde du contrevent*. Gallimard, 2007.

[Lan66] Peter J. Landin. The next 700 programming languages. *Commun. ACM*, 9(3) :157–166, 1966.

[NTT+21] Tu Dinh Ngoc, Boris Teabe, Alain Tchana, Gilles Muller, and Daniel Hagimont. Mitigating vulnerability windows with hypervisor transplant. In Antonio Barbalace, Pramod Bhatotia, Lorenzo Alvisi, and Cristian Cadar, editors, *EuroSys '21 : Sixteenth European Conference on Computer Systems, Online Event, United Kingdom, April 26-28, 2021*, pages 162–177. ACM, 2021.

3. `https://magit.vc/`

[Vik21] Vikipedio. Foino — vikipedio, 2021. [Ligite ; alirita je 8-majo-2021].