

TAS

Compact Bit Encoding Schemes for Simply-Typed Lambda-Terms

Hugo Nakagawa Jordi Bertran de Balanda

Novembre 2016

- 1** Compression de λ -termes
 - Compression d'ordre supérieur

- 2** Codage typé sans étiquettes
 - Principe
 - Transformation
 - Encodage binaire

- 3** Expérimentations

- 4** Conclusion

Compression d'ordre supérieur

- En théorie, offre un ratio de compression *très élevé*
Par exemple:
 $a^{2^n}b \equiv \text{let } twice = \lambda f. \lambda x. f(fx) \text{ in } twice^n ab$
- Produire des données compressées qui peuvent être manipulées *sans décompression*
- Émuler des schémas de compression 'classiques'

Codage typé sans étiquettes - principe

But

- 1 Retirer les étiquettes qui déterminent visuellement l'associativité des λ -termes
- 2 Générer une séquence binaire contenant informations de types et indices de De Bruijn, correspondant au programme encodé

Codage typé sans étiquettes - principe

But

- 1 Retirer les étiquettes qui déterminent visuellement l'associativité des λ -termes
- 2 Générer une séquence binaire contenant informations de types et indices de De Bruijn, correspondant au programme encodé

Outils

- 1 Environnement de typage Γ
- 2 Type du programme τ
- 3 Séquence sans étiquettes d'indices de De Bruijn

Codage typé sans étiquettes - transformation

Application

$$0(011)(011) \Rightarrow 0011011$$

Abstraction

$$(\lambda_{o \rightarrow o \rightarrow o} . \lambda : o . (10)0)0 \Rightarrow \lambda_{(o \rightarrow o \rightarrow o) \rightarrow o \rightarrow o} . \lambda_{o \rightarrow o} 1000$$

Codage typé sans étiquettes - transformation

Application

$$0(011)(011) \Rightarrow 0011011$$

Abstraction

$$(\lambda_{o \rightarrow o \rightarrow o}.\lambda : o.(10)0)0 \Rightarrow \lambda_{(o \rightarrow o \rightarrow o) \rightarrow o \rightarrow o}.\lambda_{o \rightarrow o}1000$$

Avec Γ l'environnement de types associé, on garantit qu'il est possible de retrouver la forme originale étiquetée. On définit *formellement* et on prouve les fonctions d'encodage et de décodage qui permettent de passer d'un état à l'autre.

Codage typé sans étiquettes - encodage binaire

Table de types

- Types codés en notation préfixe:
 $\llbracket o \rrbracket = 0; \llbracket \tau_1 \rightarrow \tau_2 \rrbracket = 1 \llbracket \tau_1 \rrbracket \llbracket \tau_2 \rrbracket$
- Séquence de types sans séparateurs, indexés dans leur ordre de rencontre

Codage typé sans étiquettes - encodage binaire

Table de types

- Types codés en notation préfixe:
 $\llbracket o \rrbracket = 0; \llbracket \tau_1 \rightarrow \tau_2 \rrbracket = 1 \llbracket \tau_1 \rrbracket \llbracket \tau_2 \rrbracket$
- Séquence de types sans séparateurs, indexés dans leur ordre de rencontre

Environnement, type du programme

- Types codés avec la même notation
- Séquence de types sans préfixe
- Éventuellement à recompresser 'classiquement'

Codage typé sans étiquettes - encodage binaire

Séquence de symboles

- Chaque λ_τ est remplacé par l'indice de τ , qu'on a préparé dans la table de types
- Chaque indice de variable est incrémenté de k , avec $k = 1$ l'indice le plus élevé de l'environnement
- On peut optimiser la quantité de bits utilisés en raisonnant sur les types disponibles

On encode ce résultat en binaire, donnant le résultat final.

Expérimentations

Méthodologie

- Données brutes tirées de séquençage ADN, dictionnaires, articles et jeux de test XML
- Génération de λ -termes avec un compresseur d'ordre supérieur

Expérimentations

Méthodologie

- Données brutes tirées de séquençage ADN, dictionnaires, articles et jeux de test XML
 - Génération de λ -termes avec un compresseur d'ordre supérieur
-
- Performances décevantes sur des petits jeux de tests dues à l'overhead d'informations de types
 - Performances globalement supérieures aux algorithmes précédemment disponibles

Conclusion

- Deux techniques de compression de λ -expressions
 - Sans étiquette : mieux adaptée aux petits programmes
 - Avec CFG : optimisée pour les expressions d'ordre supérieur
- Excellents résultats sur des programmes longs
- Meilleurs résultats en compression de λ -expression d'ordre supérieur
- Perspectives : adaptation à d'autres types de structures