Bab 10

Kesimpulan dan Saran

Bab ini berisi kesimpulan dan saran yang dihasilkan dari tugas akhir ini.

10.1 Kesimpulan

Berikut adalah kesimpulan dari tugas akhir ini:

1. Algoritma *Principal Type* (PT) dan algoritma pencarian *Type Inhabitant* berhasil diimplementasikan dengan PROLOG. Hasil implementasi tersebut bekerja sesuai dengan definisi algoritmanya. Dari hasil uji coba, rata-rata waktu komputasi algoritma PT adalah 78,24 *millisecond*, dan untuk algoritma pencarian *Type Inhabitant* adalah 62,61 *millisecond*.

2. Telah dipelajari literatur mengenai algoritma *Principal Type* dan algoritma pencarian *Type Inhabitant* pada $TA_{\boldsymbol{\lambda}}$, beserta konsep-konsep terkait yang diperlukan untuk memahami kedua algoritma tersebut. Selain itu, kedua algoritma tersebut juga telah dipaparkan dalam laporan tugas akhir ini dengan cara yang diharapkan dapat lebih mudah dimengerti dibandingkan dengan literatur yang sudah ada. Salah satu cara yang digunakan adalah dengan memberikan contoh penerapan algoritma untuk setiap langkahnya.

3. *Parser* untuk $\boldsymbol{\lambda}$-*term* dan tipe dari $\boldsymbol{\lambda}$-*term* berhasil dikembangkan dengan menggunakan *Definite Clause Grammar* (DCG) dalam PROLOG. *Parser* yang dikembangkan dapat berfungsi dengan baik untuk menguraikan $\boldsymbol{\lambda}$-*term* dan tipe sesuai dengan definisinya.

4. DCG memang memudahkan pembuatan *parser*.

5. Telah dikembangkan antarmuka grafis yang dapat memudahkan interaksi pengguna dengan program hasil implementasi kedua algoritma tersebut. Hal ini dicapai dengan rancangan antarmuka yang dapat meminimalisasi kesalahan *input* dari pengguna, menggunakan simbol yang representatif, dan juga mengikuti beberapa aturan *8 Golden Rules* dalam mendesain *user interface*, yaitu pencegahan kesalahan, pengurangan beban ingatan jangka pendek pengguna, mempertahankan konsistensi, memberikan umpan balik yang informatif, dan menjadikan pengguna sebagai inisiator dari aktifitas, bukan sebagai responden.

6. Antarmuka yang dikembangkan masih terdapat kekurangan. Salah satunya adalah performa tampilan yang agak lambat. Hal ini disebabkan karena *output* dari PROLOG ditampilkan oleh Java melalui perantara berkas PDF. Proses membaca berkas PDF inilah yang menyebabkan kinerja program kurang baik. Akan tetapi, penggunaan PDF sebagai perantara memudahkan proses pengembangan. Selain itu, dalam fitur penyimpanan *output* juga dipilih format PDF sebagai berkas penyimpan *output* dikarenakan portabilitas dari format PDF (*Portable Document Format*).

7. Eksplorasi terhadap algoritma *Principal Type* dalam penelitian ini membuahkan ide mengenai peningkatan efisiensi. Efisiensi tersebut dihasilkan dari penggabungan kasus II dan III pada algoritma PT. Penggabungan ini menyebabkan pengurangan sebuah aktifitas dari algoritma tersebut, yaitu aktifitas untuk mengecek apakah untuk $\boldsymbol{x}$ dari suatu abstraksi $\boldsymbol{\lambda x.P}$ berlaku $\boldsymbol{x \in FV(P)}$ atau $\boldsymbol{x \notin FV(P)}$.

8. *Software* ini bisa digunakan sebagai *test bed* untuk mempelajari TA, maupun untuk bereksperimen dalam *type theory*.

10.2 Saran

Berikut adalah beberapa saran untuk pengembangan penelitian ini:

1. Menggabungkan proses *parsing* untuk *input* dan komputasi algoritma, baik pada algoritma *Principal Type* dan algoritma pencarian *Type Inhabitant*. Hal ini diperkirakan dapat meningkatkan performa program.

2. Algoritma yang dijelaskan dalam laporan tugas akhir ini masih memiliki banyak kemungkinan untuk dilakukan efisiensi. Oleh karena itu dalam pengembangan penelitian ini, masih mungkin dilakukan efisiensi kinerja. Sebagai contoh, dalam algoritma *Principal Type*, salah satu kesempatan untuk melakukan efisiensi kinerja adalah dengan menggunakan algoritma unifikasi lain yang lebih efisien.

3. Kualitas antarmuka yang dikembangkan dalam penelitian ini masih dapat ditingkatkan, karena masih terdapat keterbatasan dan kelemahan, terutama performa saat menampilkan *output* hasil pemrosesan. Salah satu penyebab kurang baiknya performa aplikasi dalam hal tersebut adalah karena *output* menggunakan berkas PDF. Hal ini membuat program membutuhkan waktu ekstra untuk menampilkan hasil pemrosesan. Salah satu cara yang dapat ditempuh untuk mengatasi permasalahan ini adalah dengan mengembangkan algoritma *output formatting* untuk menampilkan hasil pemrosesan langsung, tanpa harus menggunakan perantara berkas PDF.