

LinguSQL Manual

Versi 0.3

Tim RUTI

Fakultas Ilmu Komputer

Universitas Indonesia

LinguSQL

RuTi-AGL Fasilkom UI Team

Reaserch Team :

- Heru Suhartanto (*Principle Investigator*)
- L. Y. Stefanus
- Belawati Wijaya
- Siti Aminah
- Ade Azurat
- Jimmy

Program Team v0.1 :

(January – June 2005)

- Rikky Wenang
- Sirajuddin Maizir
- Budiono Wibowoss

Program Team v0.2 :

(July – December 2005)

- Carroline D. Puspa
- Theresia Budiyaniti
- Slamet

Program Team v0.3 :

(August – September 2007)

- Nadissa Chairany
- Lucia Roly Prihandini
- Irvan Ferdiansyah

Pengantar

LinguSQL adalah sebuah *tool* untuk melakukan proses verifikasi dan transformasi sebuah skrip Lingu.

Apa itu Lingu ? Lingu adalah sebuah bahasa pemrograman abstrak yang dikembangkan dalam riset kerjasama antara Universitas Indonesia dengan Universitas Utrecht Belanda. Lingu dikhususkan secara spesifik untuk pengembangan aplikasi yang sangat bergantung pada operasi *database*. Istilah bahasa abstrak yang digunakan pada konteks bahwa Lingu memiliki tingkat kompleksitas dan fungsionalitas yang lebih sederhana dibandingkan dengan sebuah bahasa pemrograman umum, misalnya C,Java, dll. Pada tahapan implementasinya, bahasa abstrak ini akan ditransformasikan menjadi sebuah bahasa konkrit yang akan digunakan pada proyek tersebut, misalkan C,Java, dsb. Hal ini memberikan keleluasaan untuk melakukan *porting* dari sebuah platform ke platform lainnya.

Konsep yang mendasari Lingu adalah sebuah visi akan pengembangan perangkat lunak yang efisien dan memiliki tingkat keterjaminan kualitas yang lebih tinggi dibandingkan dengan proses-proses pengembangan yang telah ada saat ini. Lingu merupakan sebuah bahasa abstrak yang unik yang menggabungkan konsep verifikasi program dan ujicoba (*testing*) secara terintegrasi. Penggabungan kedua buah metode pengujian untuk perangkat lunak ini akan lebih mempunyai keterjaminan kualitas perangkat lunak jika dibandingkan dengan proses yang hanya memakai salah satu saja.

Proses verifikasi yang digunakan oleh Lingu adalah dengan menggunakan sebuah *theorem prover*. Skrip Lingu memiliki sebuah bagian khusus yang berisi spesifikasi dari program yang akan dijalankan. Spesifikasi-spesifikasi ini dibuat sesuai dengan bagaimana sistem yang dikembangkan akan digunakan. Spesifikasi yang dihasilkan pada skrip ini diberikan ke *theorem prover* dan kemudian akan diproses apakah valid atau tidak. Dalam kondisi spesifikasi yang diujikan tidak dapat dipecahkan oleh *theorem prover*, LinguSQL akan meminta bantuan *expert assistance* dalam memecahkan spesifikasi tersebut.

Disamping proses pengujian *whitebox* yang diwakilkan oleh proses verifikasi, Lingu juga akan diuji secara *black-box*. Pengujian ini dilakukan dengan mendefinisikan skenario-skenario untuk program yang dikembangkan. Skenario yang dihasilkan akan diujikan pada sebuah *database* yang berisi set data asli. Data ini di-*generate* secara otomatis oleh *automatic test generator* yang terdapat dalam LinguSQL.

Proses transformasi ke bahasa konkrit akan dilakukan setelah kedua proses verifikasi dan ujicoba dilakukan dengan sukses. Proses ini mengubah skrip Lingu, yang merupakan bahasa abstrak, menjadi sebuah bahasa konkrit. Pada versi ini LinguSQL hanya dapat melakukan transformasi ke dalam bahasa Java. Hasil dari transformasi ini kemudian diimplementasikan untuk *live-run* di lingkungan kerja organisasi tersebut.

Instalasi LinguSQL

LinguSQL dikembangkan dengan menggunakan Java. Selain itu, LinguSQL memerlukan dukungan dari aplikasi-aplikasi lain yang digunakan dalam proses verifikasi dan pengujian. Aplikasi lain yang dibutuhkan oleh LinguSQL adalah hol sebagai *theorem prover engine*, dan PostgreSQL sebagai *platform database* yang digunakan untuk ujicoba.

Sumber LinguSQL

Aplikasi dari LinguSQL dapat diambil dari situs-situs sebagai berikut :

Binary

Versi binary dari LinguSQL dapat diambil di <http://rooti.cs.ui.ac.id>

Source

Versi source code dari LinguSQL dapat diambil di <http://rooti.cs.ui.ac.id>, atau dapat juga diakses dengan menggunakan Subversion client pada `svn://rooti/LinguSQL/trunk/`.

Kebutuhan Sistem

LinguSQL dikembangkan dan dijalankan diatas platform Debian GNU/Linux 3.0. Dan untuk versi 0.3 telah diujicobakan pada Windows Vista & Windows XP.

Spesifikasi server pengembangan yang digunakan adalah IBM PC dengan processor Dual Intel Xeon 2.8 Ghz, memori 2 x 512 MB, dan harddisk Seagate ST40014A 40 GB 7200 rpm.

Untuk menjalankan aplikasi binari dari LinguSQL diperlukan Java JDK 1.6.0, postgresQL, HOL, dan MosML terinstall secara penuh.

Sedangkan untuk membangun source code LinguSQL, diperlukan aplikasi Java JDK 1.6.0, postgresQL, HOL, MosML, Netbeans, dan JavaCC.

Berikut url-url yang dapat digunakan untuk mengambil file-file yang diperlukan dan cara penginstallan pada sistem Linux.

Java JDK 6	: http://java.com/en/download/index.jsp .
PostgreSQL	: http://www.postgresql.org/download/ .
HOL	: http://hol.sourceforge.net/ .
MoscowML	: http://www.dina.dk/~sestoft/mosml.html .
Netbeans	: http://www.netbeans.org/downloads/
XAMPP	: http://www.apachefriends.org/download.php

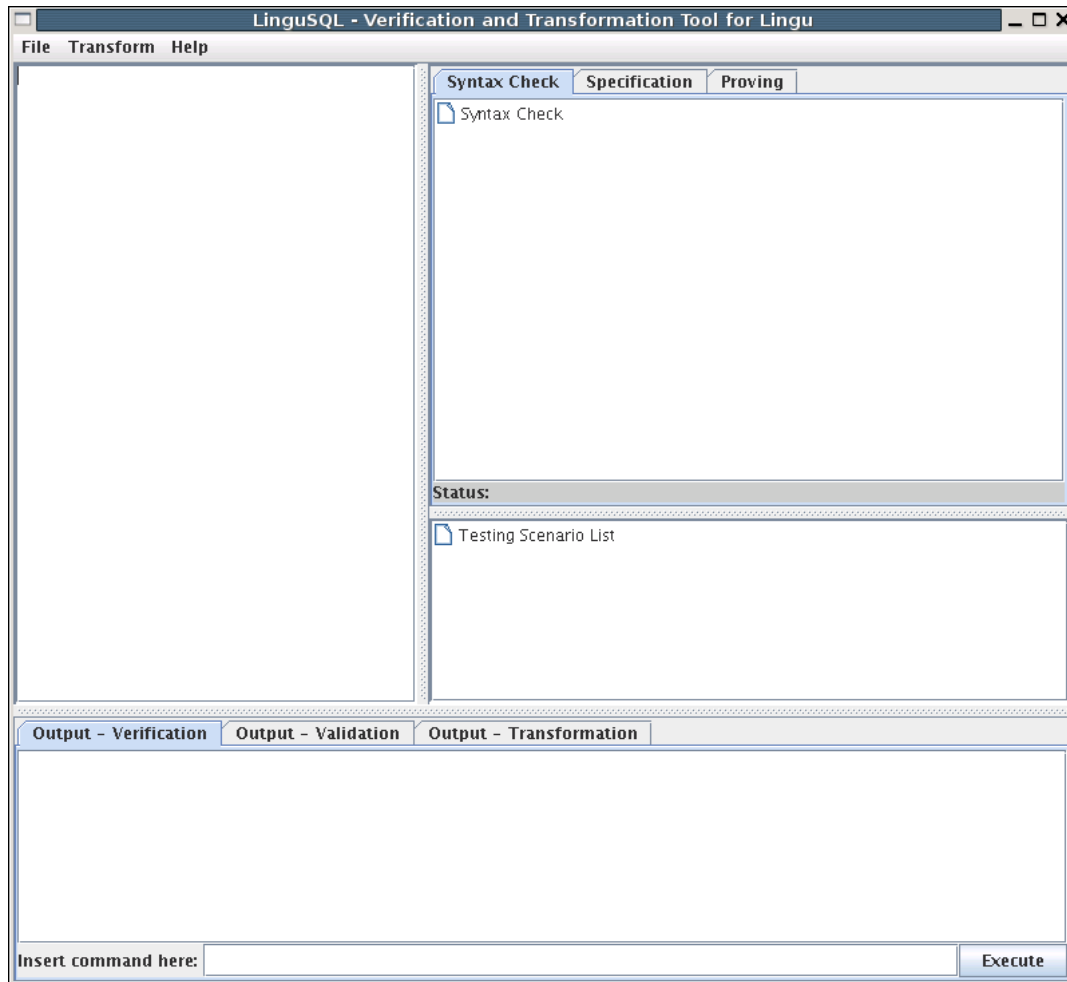
Sebelum menginstall, harus dipastikan partisi hard disk Windows anda haruslah NTFS. Manual ini ditujukan untuk Windows XP NTFS dan ikuti prosedur instalasi seperti yang tertera di dalam file Installation Awal.

Distribusi binari dari LinguSQL hanya terdiri dari sebuah file jar, yaitu **LinguSQL.jar**. Program ini

tidak perlu di-*install* secara mandiri, namun yang perlu diperhatikan adalah bahwa aplikasi-aplikasi yang dibutuhkan seperti yang disebutkan diatas harus ter-*install* dengan benar.

Sebelum menjalankan program LinguSQL pastikan bahwa classpath dan path dari java sudah ter-*setting* dengan benar. Langkah selanjutnya adalah jalankan **LinguSQL.jar** dari folder-nya dengan perintah “**java -jar LinguSQL.jar**”.

Jika perintah tersebut sukses dieksekusi maka akan muncul tampilan utama dari LinguSQL sebagai berikut :



Gambar 1 Tampilan Utama LinguSQL

Penggunaan LinguSQL

LinguSQL adalah program untuk melakukan proses verifikasi dan transformasi sebuah *script* Lingu. Untuk melakukan fungsi ini, LinguSQL membutuhkan 3 buah file. Misalkan file dari Lingu yang akan diproses bernama SET, maka harus terdapat file-file sebagai berikut :

1. **SET.lingu**, File utama ini berisi *script* pemrograman yang hendak diverifikasi dan ditransformasikan dengan menggunakan LinguSQL.
2. **SET.sce**, File ini berisi skenario yang digunakan untuk melakukan ujicoba (testing) terhadap database.
3. **SET.smx**, File ini berisi *script* HOL yang berisi spesifikasi dan *proof-script* untuk digunakan dalam proses verifikasi di HOL *theorem prover*.

Langkah-langkah yang dilakukan untuk melakukan proses LinguSQL adalah :

1. Ujicoba (*testing*) Skenario, yakni proses melakukan testing terhadap database.
2. Verifikasi script Lingu, yakni proses verifikasi terhadap *theorem prover*.
3. Transformasi, yakni transformasi ke bahasa konkrit.

Ujicoba Skenario

Ujicoba skenario (*Scenario Testing*) dilakukan untuk mengujicobakan script Lingu secara *blackbox*. tahap ini mengecek apakah output yang dihasilkan oleh script Lingu sesuai dengan yang diharapkan *programmer*. Ujicoba dilakukan terhadap *live database*.

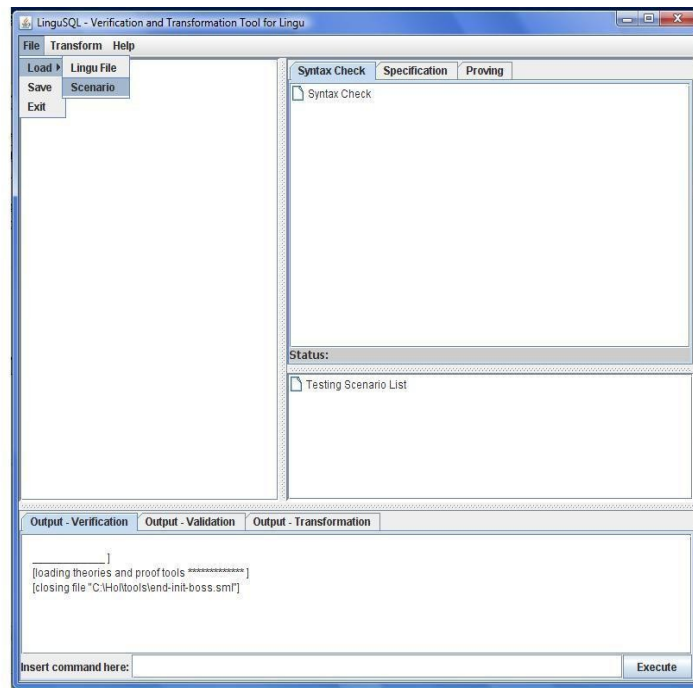
Untuk melakukan proses ini, file yang digunakan adalah file dengan extension *.sce*. File ini berisi daftar skenario yang akan dijalankan. Setiap skenario berisi urutan method / fungsi yang akan dijalankan lengkap dengan parameter-parameter yang akan diujikan. Berikut adalah contoh sebuah file skenario dengan nama *test.sce*.

```
scenario {
    test1(0,currentDate());
    test1(#a.Main-1,currentDate());
    test2(0,currentDate());
    test1(#a.Main, currentDate());
}
```

Tabel 1 Contoh File Skenario

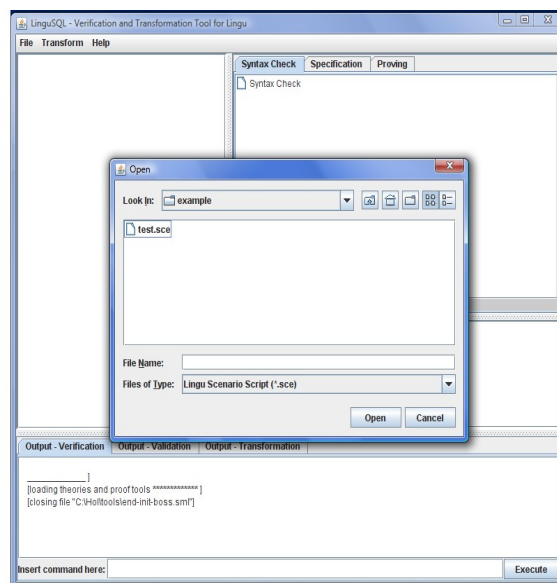
Skenario diatas berisi 4 buah test dengan nama test1 dan test2. Untuk fungsi test1, terdapat 3 buah perintah ujicoba dengan parameter yang berbeda. Sedangkan untuk test2 hanya terdapat satu buah perintah ujicoba dengan parameter 0 dan tanggal saat ini (*currentDate()*).

Untuk menjalankan ujicoba scenario, klik tombol *Load Scenario* yang terdapat di bagian atas tampilan utama LinguSQL.



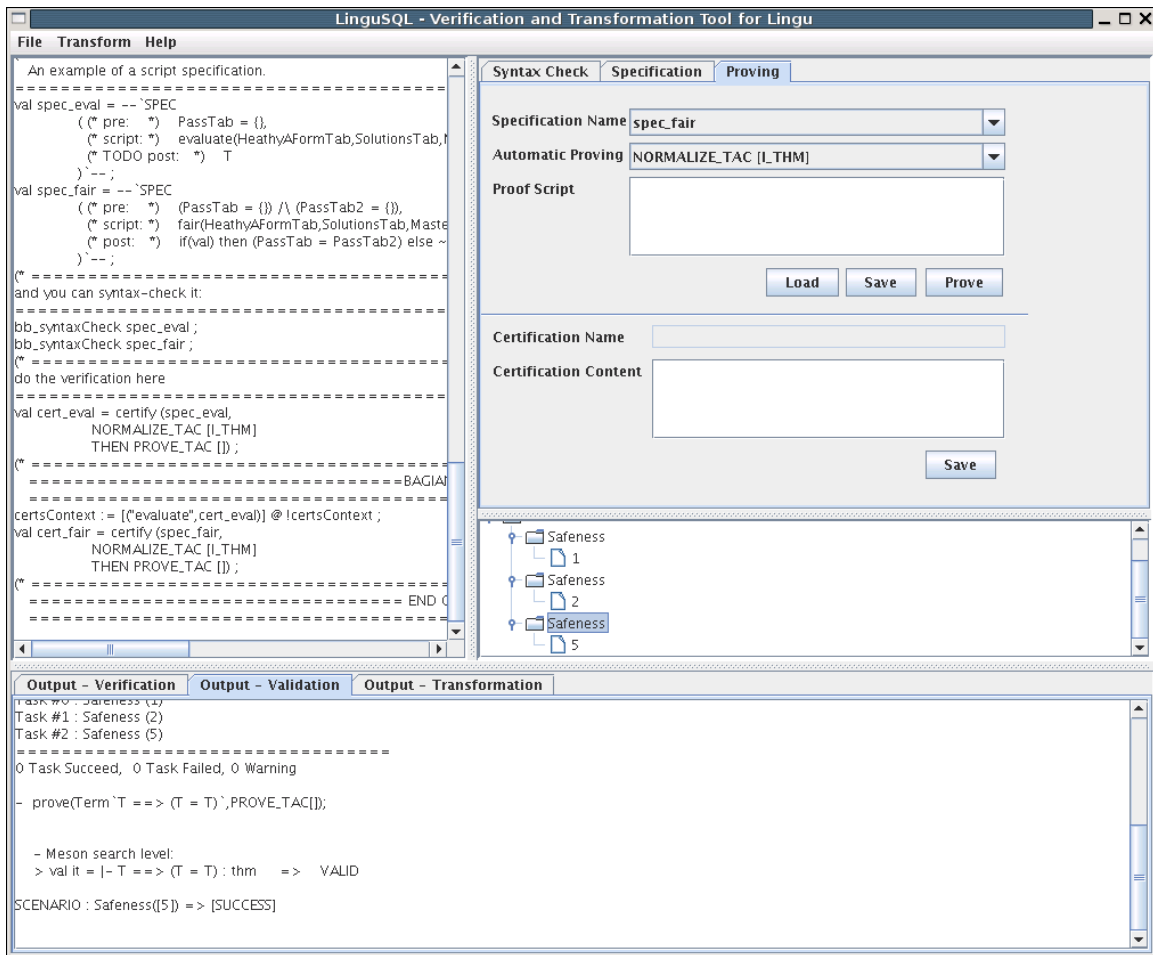
Gambar 2 Load Scenario

Setelah melakukan klik pada tombol Load Scenario, LinguSQL akan menampilkan kotak untuk memilih file skenario di komputer anda. Pilih file dengan extension *.sce* yang akan digunakan untuk ujicoba (*testing*) script Lingu.



Gambar 3 Pilih File Skenario

Jika file skenario sukses dibuka, maka tampilan pada area *Lingu Testing* akan berisi daftar skenario dan perintah ujicoba yang akan dijalankan. Berikut adalah tampilan dari skenario yang berhasil dibuka.



Gambar 4 Detail Skenario

Nama Skenario adalah nama-nama dari skenario yang terdapat di file skenario Lingu. Tiap skenario memiliki satu set perintah yang akan dijalankan untuk ujicoba. Perintah tersebut di tampilkan pada *Daftar Perintah Ujicoba*. Jika salah satu skenario diklik, maka aplikasi akan menampilkan detail dari skenario tersebut. Detail yang terdapat pada *Detail Skenario* ini berisi daftar perintah yang akan dijalankan beserta parameternya, serta status keberhasilan dari masing-masing ujicoba.

Verifikasi Lingu Script

Tahap selanjutnya setelah ujicoba adalah melakukan verifikasi dari skrip Lingu. verifikasi dilakukan dengan menganalisa script Lingu dan memecahnya menjadi *verification conditions* yang kemudian akan dilempar ke HOL untuk diperiksa kebenarannya.

File yang diperlukan untuk melakukan proses verifikasi adalah file dengan extension *.smx*. File ini berisi script Lingu yang telah diubah kedalam bentuk HOL dan siap untuk diverifikasi. File ini terdiri dari 5 bagian, yaitu deklarasi *library*, deklarasi *datatype*, fungsi, deklarasi spesifikasi, dan *proof script*.

1. Deklarasi library adalah deklarasi yang dibuat untuk membuka dan memanggil library yang akan digunakan untuk melakukan proses verifikasi. Contoh dari bagian tersebut adalah sebagai berikut.

```

(* HOL Libraries *)
load "intLib" ;
load "stringLib" ;
load "pred_setLib" ;
load "combinTheory";
open intLib stringLib pred_setLib combinTheory;

(* Lingu Specific *)
load "/home/wenang/ruti/workspace/lingux" ;
open /home/wenang/ruti/workspace/lingux;

```

2. Deklarasi datatype adalah deklarasi dari struktur data yang akan digunakan oleh skrip Lingu pada proses komputasi. Contoh dari bagian ini adalah sebagai berikut.

```

val _ = Hol_datatype `RegistrationTable =
  <| ID      : string ;
     Name   : string ;
     Sex    : int;
     Category : int;
     StudyProgramme : string |> ` ;

val _ = Hol_datatype `AnswerFormTable =
  <| ID      : string ;
     Name   : string ;
     SheetCode : string ;
     Answer : string |> ` ;

val _ = Hol_datatype `SolutionTable =
  <| Answer : string |> ` ;

```

3. Fungsi adalah deklarasi dari alur fungsi yang akan digunakan. Fungsi ini yang berisi proses bisnis dan logika dari aplikasi database.

```

val fair_def = Define
  `fair ( HealthyAFormTab : AnswerFormTable set,
          SolutionsTab : SolutionTable set,
          MasterTab : RegistrationTable set,
          PassTab : RegistrationTable set,
          PassTab2 : RegistrationTable set,
          valid : bool
        ) (dummy) =
  call (evaluate (HealthyAFormTab, SolutionsTab, MasterTab,
                PassTab) (dummy))
  >>
  update MasterTab (\r. r with Sex := 0 ) (\c. ~(c.Sex = 0))
  >>
  call (evaluate (HealthyAFormTab, SolutionsTab, MasterTab,
                PassTab2) (dummy))
  >>
  if PassTab = PassTab2
  then valid ::= T else valid ::= F `;

```

4. Deklarasi spesifikasi adalah deklarasi dari spesifikasi yang dibutuhkan dan harus dipenuhi oleh sebuah fungsi. Spesifikasi mendeskripsikan nilai yang harus dipenuhi oleh sebuah fungsi sebelum dimulai (*preconditions*), setelah eksekusi (*postconditions*), dan nilai akhir (*return value*). Contoh dari spesifikasi adalah sebagai berikut.

```

val spec_fair = --`SPEC (
(* pre: *)      (PassTab = {}) /\ (PassTab2 = {}),
(* script:*)   fair (HeathyAFormTab,SolutionsTab,MasterTab,PassTab,
                    PassTab2,val)(dummy),
(* post: *)    if(val) then (PassTab = PassTab2) else ~(PassTab =
                    PassTab2) )`-- ;

```

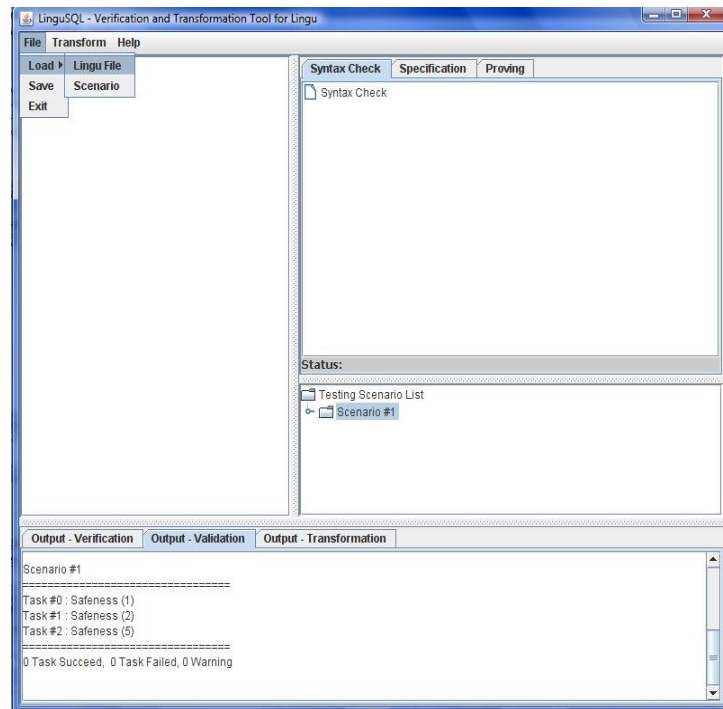
5. *Proof script* adalah langkah-langkah pembuktian yang akan digunakan untuk mem-verifikasi fungsi dan spesifikasi yang telah dideskripsikan sebelumnya. Langkah-langkah yang untuk mem-verifikasi ini diambil dari library HOL *theorem prover*. Contoh dari bagian ini adalah sebagai berikut.

```

val cert_fair = certify (spec_fair,
                        NORMALIZE_TAC [I_THM]
                        THEN PROVE_TAC []) ;

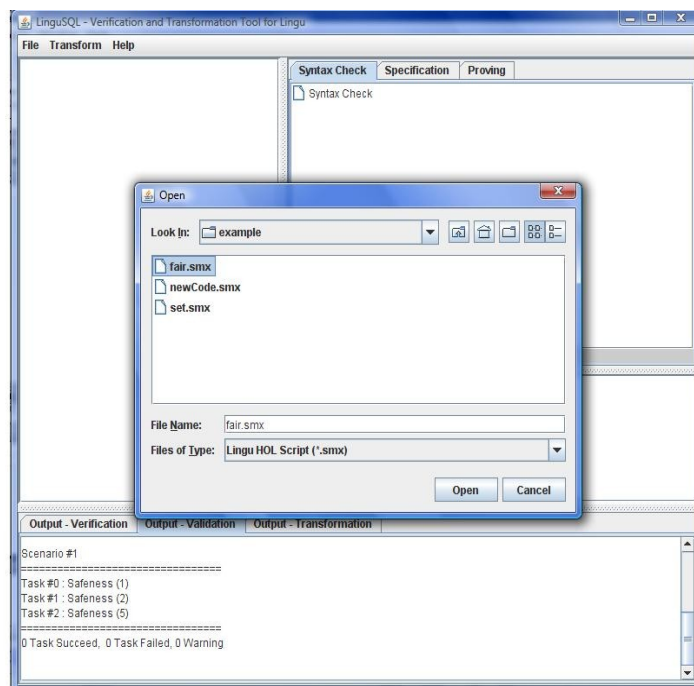
```

Untuk memulai proses verifikasi, klik tombol *Load Lingu* pada tampilan utama.



Gambar 5 Tombol Load Lingu

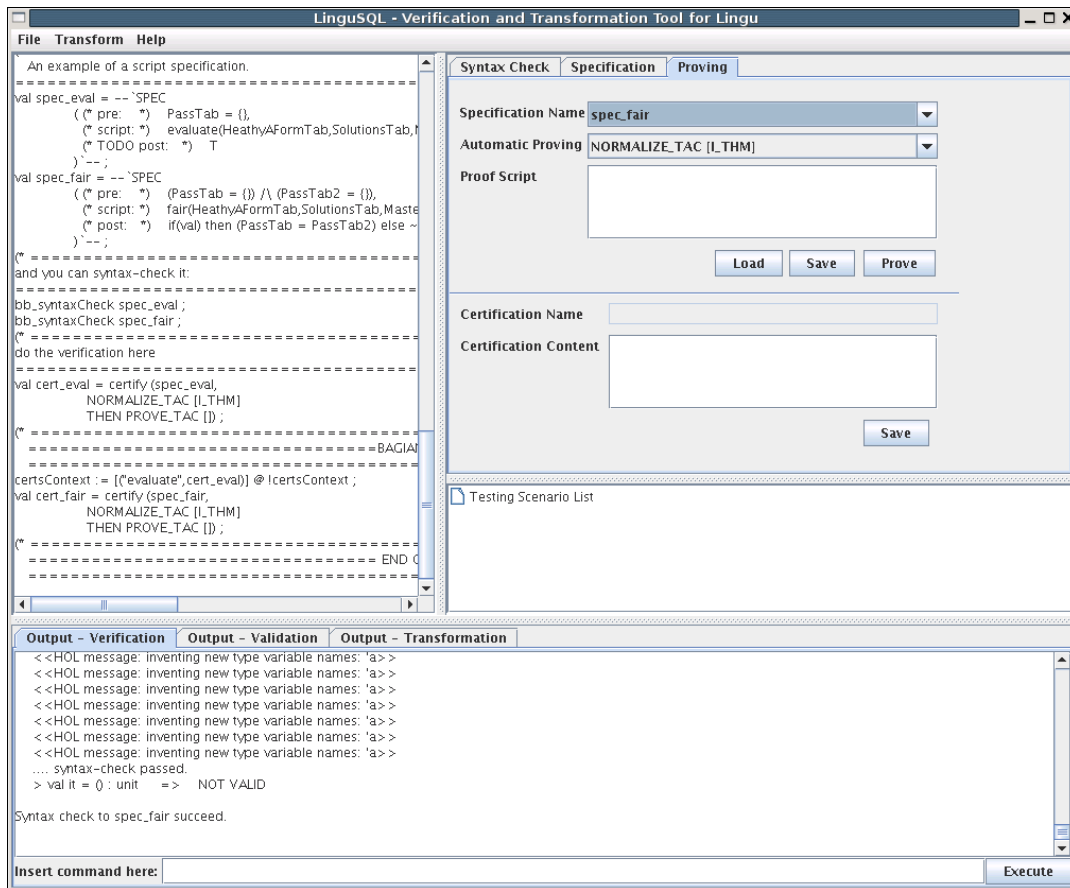
Tombol ini setelah di-klik akan menampilkan pilihan file yang akan digunakan dalam proses verifikasi ini. Contoh tampilannya adalah sebagai berikut.



Gambar 6 Pilih File LinguHOL

Dari contoh gambar 7 setelah memilih sebuah file smx, maka aplikasi akan membuka file tersebut dan

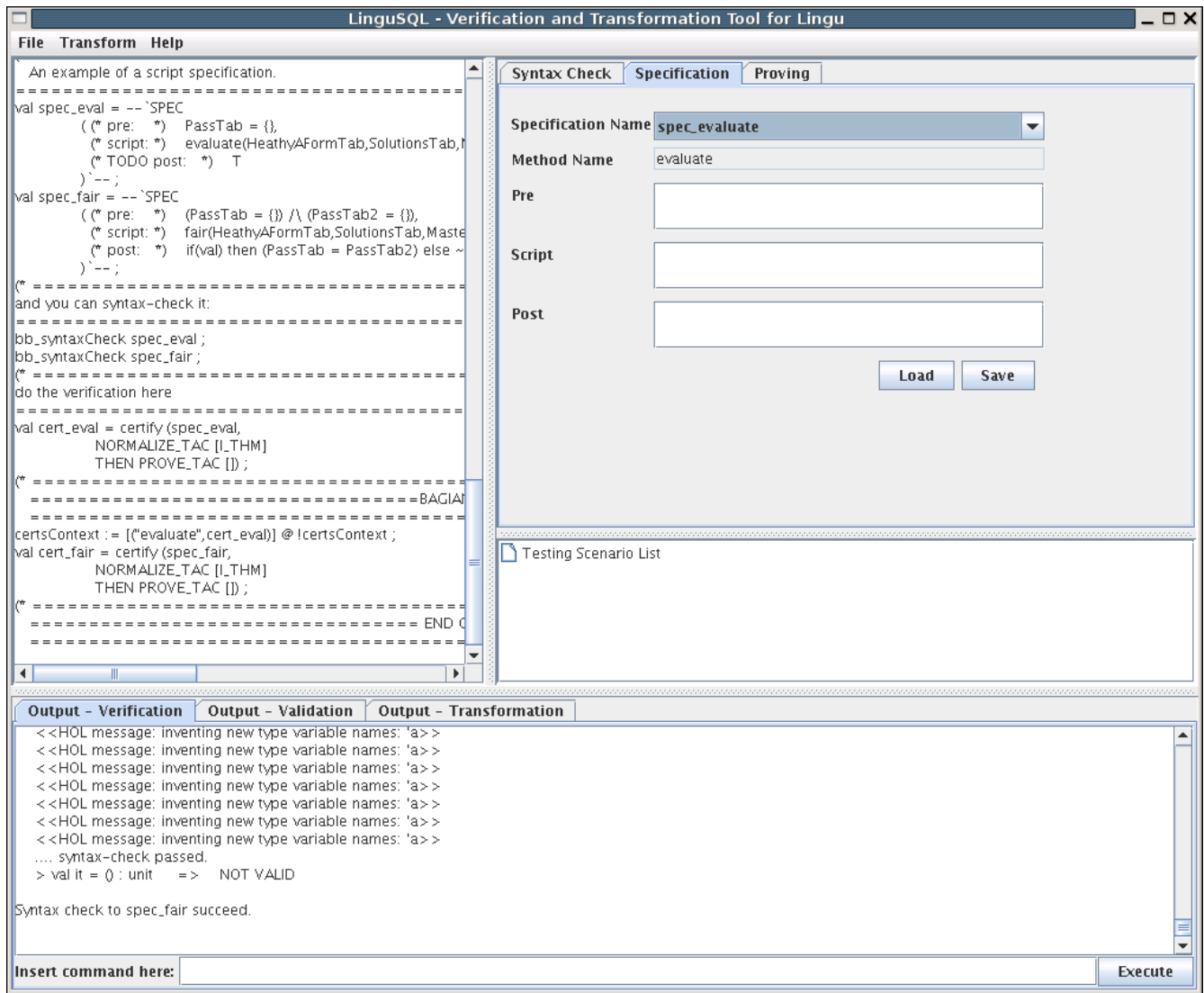
melemparkannya ke HOL *theorem prover*. Hasil yang didapat dari *theorem prover* tersebut akan ditampilkan pada kotak verifikasi di bagian kanan. Hasil ini berisi pembuktian dan menampilkan apakah skrip Lingu yang diverifikasikan memenuhi syarat atau tidak. Berikut adalah contoh tampilannya.



Gambar 7 Hasil Verifikasi

Sifat dari verifikasi dengan HOL adalah *undecidable*. Artinya adalah bahwa proses ini tidak bisa 100% secara otomatis menangani pembuktian yang diperlukan oleh sebuah skrip. Pembuktian matematis secara otomatis masih belum bisa mencakup semua permasalahan. Karenanya terkadang dibutuhkan bantuan manusia untuk memberitahu *theorem prover* mengenai formula apa saja yang harus digunakan untuk memecahkan masalah tersebut. Sifat *undecidable* ini menyebabkan aplikasi LinguSQL menyediakan sebuah media untuk memecahkan masalah secara manual. Di aplikasi LinguSQL disediakan sebuah *box* untuk memasukkan formula secara manual apabila HOL mengalami sebuah masalah dalam memecahkan sebuah formula. Contohnya adalah sebagai berikut.

Command box yang terdapat di LinguSQL dapat menerima perintah-perintah spesifik HOL untuk memecahkan masalah.



Proses verifikasi selesai apabila seluruh script telah terbukti oleh HOL *theorem prover*. Dengan menggunakan metode ini, seorang programmer telah melakukan *whitebox testing* terhadap kodenya. Metode ini telah teruji mampu menekan jumlah error yang muncul selama pengembangan jika dibandingkan dengan metode *blackbox* saja. Setelah programmer yakin bahwa kode yang dibuat di bahasa abstrak ini telah teruji, maka sudah saatnya untuk melakukan transformasi dari Lingu ke bahasa konkrit yang akan digunakan pada organisasi.