PIRANTI LUNAK PEMBUKTIAN PERNYATAAN LOGIKA PROPOSISI DENGAN METODE RESOLUSI MENGGUNAKAN BAHASA PEMROGRAMAN PROSEDURAL

Arnold Aribowo, Kristian Frits Harris, Budi Berlinton Sitorus
Universitas Pelita Harapan, Indonesia
UPH Tower, Lippo Karawaci, Tangerang 15811, Indonesia
E-mail: arnold@uph.edu, budibs@uph.ac.id

1. ABSTRACT

Proving by resolution method is one of methods which can be used to prove the logical statement according to given statements. To automate the proving process and to support the teaching-learning process of discrete mathematics course in the university, this proving can be done with the aid of computer software. To enable the effective use of software, the error handler facilities are added. They help users to overcome erroneous inputs and therefore facilitate the use of software.

In this paper, several possibilities of mistaken input performed by users are elaborated. By analyzing them, the error handler facilities are added in the software for proving the propositional logic statement by using resolution method. According to the testing which was performed, the error handler facilities can be applied in the software.

Keywords: *Resolution, Conjunctive Normal Form, Automated Reasoning, Handling Errors.*

1. PENDAHULUAN

Salah satu metode yang dapat digunakan dalam pembuktikan kebenaran pernyataan logika proposisi adalah metode resolusi. Secara umum, pembuktian dengan metode resolusi pada logika proposisi terdiri dari 3 langkah, yaitu pengubahan proposisi asumsi ke dalam bentuk *Conjunctive Normal Form* (CNF), negasi proposisi yang hendak dibuktikan dan pengubahan proposisi ke dalam bentuk CNF (jika diperlukan), serta pembuktian proposisi dalam bentuk CNF dengan metode resolusi.

Proses pembuktian tersebut pada umumnya dilakukan secara manual. Bagi sebagian mahasiswa yang terlibat dalam proses belajar mengajar, proses pembuktian ini tidak mudah dipahami, terutama pada persoalan-persoalan kompleks. Untuk mengatasi masalah tersebut, piranti lunak untuk melakukan pembuktian dengan metode resolusi secara otomatis telah dikembangkan.

Ketika piranti lunak digunakan oleh pengguna, terdapat beberapa kemungkinan kesalahan yang dilakukan oleh pengguna dalam pemasukkan data. Untuk mengatasinya, pada piranti lunak yang dikembangkan tersebut ditambahkan fasilitas penanganan kesalahan. Selain membahas pembuktian dengan metode resolusi dengan bantuan piranti lunak, makalah ini juga membahas mengenai kemungkinan kesalahan masukan pengguna dan selanjutnya menunjukkan bahwa pada piranti lunak yang dikembangkan terdapat fasilitas penanganan kesalahan yang dapat membantu pengguna. Piranti lunak ini dikembangkan menggunakan bahasa pemrograman prosedural, C++. Piranti lunak serupa juga telah dibuat, tetapi menggunakan bahasa pemrograman deklaratif, yaitu Prolog [2].

2. LANDASAN TEORI

Teori pendukung yang mendasari penelitian yang dilakukan meliputi : Proposisi dan *Operator* logika, Formula, CNF, dan Pembuktian dengan Metode Resolusi.

2.1. PROPOSISI DAN *OPERATOR* LOGIKA

Logika proposisi adalah logika yang didasarkan pada proposisi. Sebuah proposisi adalah sebuah pernyataan yang berisi nilai kebenaran `True` atau `False`, tapi tidak keduanya [4]. Dengan demikian dapat dikatakan bahwa nilai kebenaran (*truth value*) dari suatu proposisi adalah `True` (T) atau `False` (F). Sebuah pernyataan proposisi hanya dapat mempunyai salah satu dari dua nilai kebenaran tersebut. Selain proposisi yang mempunyai nilai kebenaran yang tetap, ada nilai kebenaran proposisi yang dapat berubah tergantung dari situasi [5]. Walaupun nilai kebenaran sebuah proposisi dapat berubah tergantung situasi, suatu proposisi tidak pernah bernilai benar dan salah dalam waktu yang bersamaan.

Dalam logika proposisi terdapat beberapa *operator* logika. *Operator* logika dibagi menjadi 2 jenis *operator*: *unary* dan *binary*. Negasi termasuk *unary operator*, sedangkan konjungsi, disjungsi, implikasi, dan ekivalensi termasuk *binary operator* [5].

2.2. FORMULA

Proposisi yang tersusun dengan benar disebut *well-formed formula* atau formula [3]. Formula, dalam logika proposisi dapat didefinisikan secara rekursif sebagai berikut:

- Sebuah atom adalah sebuah formula.
- Apabila G merupakan sebuah formula, maka $\neg$ G merupakan formula.
- Apabila G dan H merupakan formula, maka (G $\wedge$ H), (G $\vee$ H), (G $\rightarrow$ H), dan (G $\leftrightarrow$ H) adalah formula.
- Semua formula dapat dihasilkan dengan menerapkan aturan-aturan di atas.

2.3. *CONJUNCTIVE NORMAL FORM* (CNF)

Sebuah *literal* merupakan sebuah atom atau negasi dari atom [3]. Sebuah formula F dikatakan dalam bentuk CNF jika dan hanya jika formula tersebut memiliki bentuk $F = F_1 \wedge F_2 \wedge F_n, n \geq 1$, dimana $F_1, \ldots, F_n$ merupakan disjungsi dari *literal*. Apabila diketahui p, q, dan r merupakan atom, maka $F = (p \vee \neg q \vee r) \wedge (\neg p \vee q)$ merupakan formula dalam bentuk CNF. Berikut ini adalah langkah-langkah yang harus dilakukan untuk mengubah sebuah formula ke dalam bentuk CNF:

a) Menghilangkan *operator* $\leftrightarrow$ dan $\rightarrow$ dengan formula yang ekivalen, yaitu :
   $p \leftrightarrow q \Leftrightarrow p \rightarrow q) \wedge (q \rightarrow p)$
   $p \rightarrow q \Leftrightarrow \neg p \vee q$
b) Mengurangi scope tanda $\neg$ dengan menggunakan *De Morgan's laws* dan *double negation rule* :
   $\neg(\neg(p)) \Leftrightarrow p$
   $\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$
   $\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$
c) Menggunakan aturan distribusi untuk mengubah pernyataan ke dalam CNF:
   $p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$
   $(p \wedge q) \vee r \Leftrightarrow (p \vee r) \wedge (q \vee r)$

2.4. PEMBUKTIAN DENGAN METODE RESOLUSI

Apabila diketahui proposisi asumsi dalam bentuk CNF $F = F_1 \wedge F_2 \wedge F_n, n \geq 1$, dimana $F_1, \ldots, F_n$ merupakan disjungsi dari *literal*, dan diketahui G, yang merupakan proposisi yang hendak dibuktikan juga dalam bentuk CNF, G dapat dibuktikan menggunakan metode resolusi dengan langkah-langkah berikut :

a) Menegasikan proposisi (*goal*) yang hendak dibuktikan dan menambahkannya pada daftar *clauses* bersama-sama dengan *clauses* sebelumnya yang dihasilkan dari proposisi asumsi. Dengan demikian, daftar *clauses* terdiri dari : $F_1, F_2, \ldots, F_n, \sim G$.
b) Berdasarkan asumsi-asumsi yang ada, 2 clauses yang memiliki satu pasang *literal* yang komplementer dipilih, yaitu :
   F_n : $\sim P \vee Q \vee R$ dan F_m : $S \vee P$.
c) Dengan menghapus pasangan *literal* yang komplementer tersebut, *clause* baru dapat dihasilkan, yaitu : $Q \vee R \vee S$.
d). Ulangi langkah-langkah di atas hingga dihasilkan *clause* False.

3. IMPLEMENTASI PIRANTI LUNAK

Untuk melakukan pembuktian dengan metode resolusi, diperlukan data masukan berupa proposisi. Proses memasukkan proposisi dilakukan dengan *keyboard*. Untuk menyimpan data yang dimasukkan berupa proposisi, digunakan sebuah *array of character* yang diberi nama *input*. Penggunaan *array of character* dimaksudkan untuk mempermudah analisis karakter per karakter. Karakter-karakter yang valid digunakan sebagai data masukan proposisi adalah:

1) Karakter alfabet huruf kecil.
2) Karakter *operator* logika: ~, \/, /\, =>,<=>.
3) Karakter kurung buka dan kurung tutup.

Pembuktian dengan metode Resolusi memerlukan minimal sebuah proposisi yang hanya mengandung sebuah variabel yang dipisahkan oleh *operator* konjungsi. Kondisi tersebut mengindikasikan keberadaan variabel tunggal. Contoh proposisi yang mengandung variabel tunggal adalah p/\q, p/\(q\/r), (a\/b)/\(c\/d)/\e. Sebelum dilakukan pembuktian dengan menggunakan metode resolusi, proposisi yang hendak dibuktikan terlebih dahulu harus diubah ke dalam bentuk CNF dengan tahapan-tahapan yang telah dijelaskan sebelumnya. Hasil proses konversi CNF akan ditampilkan dengan menggunakan penomoran. Jika pada hasil CNF terdapat tanda kurung, maka tanda kurung tersebut akan dihilangkan terlebih dahulu sebelum ditampilkan.

Pembuktian dengan menggunakan metode resolusi dapat dilakukan sesuai dengan langkah-langkah yang telah dijelaskan sebelumnya. Jika hasil pembuktian bernilai FALSE, maka terdapat kemungkinan bahwa terdapat langkah-langkah pembuktian yang tidak memberikan kontribusi terhadap pembuktian. Dengan tidak menampilkan langkah-langkah tersebut, maka langkah-langkah pembuktian yang ditampilkan akan menjadi lebih efisien.

Untuk menangani kesalahan penulisan proposisi pada saat memasukkan data, maka dirancang beberapa penanganan kesalahan. Jenis-jenis kesalahan penulisan proposisi pada saat memasukkan data antara lain:

1) Kesalahan penulisan karakter.

Jika terdapat karakter yang tidak valid, maka akan diberikan empat kemungkinan pilihan untuk mengatasi kesalahan tersebut, yaitu:

1) Hilangkan karakter tidak valid tersebut.
2) Tukar karakter tidak dikenal dengan *operator* logika.
3) Tukar karakter tidak dikenal dengan proposisi baru.
4) Menuliskan proposisi baru.

2) Kesalahan penulisan tanda kurung.

Untuk mengatasi kesalahan ini maka pengguna diberikan pilihan untuk memilih letak tanda kurung buka ataupun tanda kurung tutup.

3) Kesalahan penulisan variabel.

Dalam menuliskan variabel untuk sebuah proposisi, terdapat dua kemungkinan kesalahan penulisan, yaitu:

1) Penulisan lebih dari satu variabel.
   Penulisan variabel harus mengacu kepada sifat *operator* logika *unary* dan *binary*. Jika ditemukan keberadaan proposisi yang memiliki lebih dari satu variabel, maka pengguna akan dihadapkan dengan empat pilihan operasi, yaitu:
   1) Memilih salah satu variabel.
   2) Menambahkan *operator* logika yang memiliki sifat *binary*.
   3) Mengganti bagian tersebut dengan proposisi baru.
   4) Memasukkan proposisi baru.
2) Penulisan variabel dengan karakter alfabet kapital.
   Variabel yang valid untuk dioperasikan oleh piranti lunak adalah karakter alfabet huruf kecil. Untuk membedakan karakter dalam data masukan, maka setiap karakter akan dilihat nilainya menurut *American Standard Code for Information Interchange* (ASCII). Nilai ASCII antara alfabet huruf kecil dengan alfabet huruf kapital memiliki selisih sebanyak 32. Untuk mengkonversi karakter alfabet huruf kapital menjadi karakter alfabet huruf kecil, maka nilai ASCII dari karakter alfabet huruf kapital ditambahkan dengan nilai 32.

4) Kesalahan penulisan *operator* logika.

Penulisan *operator* logika di dalam proposisi harus memperhatikan sifat *operator* logika dalam variabelnya. *Operator unary* hanya membutuhkan sebuah variabel, sedangkan *operator binary* membutuhkan dua buah variabel.

4. PENGUJIAN

4.1. PENGUJIAN PIRANTI LUNAK

Pengujian piranti lunak dilakukan pada 20 jenis variasi soal pembuktian pernyataan logika proposisi yang hendak dibuktikan. Berikut ini adalah salah satu contoh pengujian pada piranti lunak untuk pembuktian pernyataan logika proposisi menggunakan metode resolusi:

- Apabila diberikan data masukan proposisi `(p\/q)/\(~p\/r)/\(~q\/s)` dan hendak dibuktikan `r\/s`, proses pembuktian dapat diperlihatkan pada Gambar 1.

```
============
Konversi CNF
============

Input: (p\/q)/\(~p\/r)/\(~q\/s)


GOAL: r\/s

=================
Konversi CNF Goal
=================


===========
Negasi Goal
===========

~r/\~s

===================
Pembuktian Resolusi
===================

Asumsi:
1. p\/q
2. ~p\/r
3. ~q\/s
4. ~r
5. ~s
6. ~p              Resolution 4&2
7. ~q              Resolution 5&3
8. q               Resolution 6&1
9. FALSE           Resolution 7&8

press any key to continue . . .
```

Gambar 1. Pengujian Pembuktian dengan metode Resolusi

Berikut ini diperlihatkan tabel hasil pembuktian dengan metode Resolusi pada variasi soal yang ada :

Tabel 1. Hasil Pembuktian Dengan Metode Resolusi

Pernyataan Logika	Keterangan
(p\/q)/\(~p\/r)/\(~q\/s) |- r\/s	Berhasil
(p/\q)/\(p=>r)/\((q/\r)=>s) |- s	Berhasil
(~p\/~q\/r)/\p/\q |- r	Berhasil
((p/\q)=>r)/\(w=>r)/\p/\(s=>w)/\s |- r	Berhasil
(~p/\q/\(~p=>(q=>r))) |- r	Berhasil
p/\(p=>q)/\(p=>(q=>r)) |- r	Berhasil
((p/\~q)=>r)/\~r/\p |-q	Berhasil
~q/\(p=>q)/\(r\/p) |- r	Berhasil
((p/\q)=>r)/\p/\q |- r	Berhasil
(q=>r)/\(p\/q) |- p\/r	Berhasil
(q=>r)/\(~q=>~p) |- p=>r	Berhasil
(p=>q)/\(~p=>r)/\(r=>s) |- ~q=>s	Berhasil
(p=>(q=>r))/\p/\~r |- ~q	Berhasil
((p/\q)/\r)/\(s/\t) |- q/\s	Berhasil
(p/\q)/\r |- q/\r	Berhasil
((a=>c)/\(b=>c)) |- (a\/b)=>c	Berhasil
p/\~(~(q/\r)) |- ~(~p)/\r	Berhasil
(q=>r)/\(p\/q)/\~p |- r	Berhasil
(q=>r)/\(~q=>~p)/\p |- r	Berhasil
~c/\(i=>c)/\(a\/i) |- a	Berhasil

4.2. PENANGANAN KESALAHAN PENGGUNA

Pada halaman *input* data diperlukan sebuah data masukan berupa proposisi yang valid. Proses memasukkan data berupa proposisi diakhiri dengan tombol "*ENTER*" pada *keyboard*. Terdapat empat jenis kesalahan dalam menuliskan proposisi, yaitu:

1) Kesalahan penulisan karakter.

Apabila diberikan data masukan berupa proposisi `a.b`, proposisi tersebut mengandung kesalahan karakter tidak dikenal. Karena keberadaan karakter tidak dikenal di dalam proposisi, maka diberikan empat buah pilihan untuk memperbaiki proposisi tersebut (Gambar 2).

```
====================
Karakter Tidak Dikenal
====================

Data masukan anda:a.b

Kesalahan karakter: .

Pilihan operasi:
  1. Hilangkan karakter tidak dikenal tersebut.
  2. Tukar karakter tidak dikenal dengan operator logika.
  3. Tukar karakter tidak dikenal dengan proposisi baru.
  4. Menuliskan proposisi baru
Pilihan anda:
```

Gambar 2. Pilihan Operasi Untuk Kesalahan Karakter Tidak Dikenal

Jika memilih pilihan "Hilangkan karakter tidak dikenal tersebut", maka pengguna akan mendapatkan hasil seperti yang diperlihatkan pada Gambar berikut :

```
==========================
Perbaikan Variabel Berganda
==========================

Data masukan anda:ab

Pilihan operasi yang dapat dilakukan:
1. Memilih salah satu variabel
2. Menambahkan operator logika
3. Mengganti bagian tersebut dengan proposisi baru
4. Memasukkan proposisi baru
Pilihan anda:_
```

Gambar 3. Hasil Menghilangkan Karakter Tak Dikenal

Perbaikan variabel berganda akan dijelaskan lebih lanjut pada bagian lain makalah ini. Jika dipilih "Tukar karakter tidak dikenal dengan operator logika", maka pengguna akan diperlihatkan tampilan seperti pada Gambar berikut:

```
====================
Menambahkan Operator
====================


Input anda:ab

Pilih operator logika:
   1. Konjungsi  : /\
      a/\b
   2. Disjungsi  : \/
      a\/b
   3. Implikasi  : =>
      a=>b
   4. Ekuvalensi : <=>
      a<=>b
Pilihan anda:1

Input anda yang telah diubah:a/\b
Apakah data masukan anda telah benar?(y/n)
```

Gambar 4. Hasil Mengganti Karakter Tak Dikenal Dengan Operator Logika

Jika memilih "Tukar karakter tidak dikenal dengan proposisi baru", maka pengguna akan mendapatkan hasil sebagai berikut :

```
====================
Menyisipkan Proposisi
====================

Data masukan anda:
  a.b

Masukkan proposisi untuk disisipkan: /\c/\


Nilai data masukan yang baru:
   a/\c/\b

Apakah proposisi baru anda sudah benar?(y/n)
```

Gambar 5. Hasil Mengganti Karakter Tidak Dikenal Dengan Proposisi Baru pada Proposisi

2) Kesalahan penulisan tanda kurung.

Apabila diberikan data masukan berupa proposisi `(((a/\b)/\c/\d)/\e`, maka akan diperlihatkan pesan kesalahan berikut ini :

```
================================
Menambahkan Karakter Tanda Kurung
================================

Input anda: (((a/\b)/\c/\d)/\e
Proposisi anda kekurangan karakter tanda kurung tutup

Kemungkinan letak tanda kurung tutup:
1. (((a)/\b)/\c/\d)/\e
2. (((a/\b))/\c/\d)/\e
3. (((a/\b)/\c)/\d)/\e
4. (((a/\b)/\c/\d))/\e
5. (((a/\b)/\c/\d)/\e)
Pilihan anda:
```

Gambar 6. Kemungkinan Letak Tanda Kurung Tutup

3) Kesalahan penulisan variabel.

Apabila diberikan data masukan berupa proposisi `ab`, proposisi tersebut memiliki variabel berganda. Untuk mengatasi kesalahan penulisan variabel, maka diberikan empat pilihan operasi (Gambar 7).

```
==========================
Perbaikan Variabel Berganda
==========================

Data masukan anda:ab

Pilihan operasi yang dapat dilakukan:
1. Memilih salah satu variabel
2. Menambahkan operator logika
3. Mengganti bagian tersebut dengan proposisi baru
4. Memasukkan proposisi baru
Pilihan anda:
```

Gambar 7. Pilihan Operasi Dalam Kesalahan Penulisan Variabel

Jika pengguna memilih "Memilih salah satu variabel", maka akan diberikan tampilan sebagai berikut :

```
================
Memilih Variabel
================

Data masukan anda:ab

Pilihan variabel untuk digunakan:
1. a
   a
2. b
   b
Pilihan anda:
```

Gambar 8. Hasil Memilih Salah Satu Variabel

Jika pengguna memilih "Menambah operator logika", maka pengguna akan diberikan tampilan sebagai berikut :

```
====================
Menambahkan Operator
====================


Input anda:ab

Pilih operator logika:
   1. Konjungsi  : /\
      a/\b
   2. Disjungsi  : \/
      a\/b
   3. Implikasi  : =>
      a=>b
   4. Ekuvalensi : <=>
      a<=>b
Pilihan anda:1

Input anda yang telah diubah:a/\b
Apakah data masukan anda telah benar?(y/n)
```

Gambar 9. Hasil Pilihan Menambahkan Operator

Jika pengguna memilih "Mengganti bagian tersebut dengan proposisi baru", maka pengguna akan diberikan tampilan sebagai berikut :

```
====================
Menyisipkan Proposisi
====================

Data masukan anda:
  a/\bc/\d

Masukkan proposisi untuk disisipkan: b

Nilai data masukan yang baru:
   a/\b/\d

Apakah proposisi baru anda sudah benar?(y/n)
```

Gambar 10. Hasil Pilihan Menyisipkan Proposisi

4) Kesalahan penulisan operator logika.

a) Operator logika *unary*.

Apabila diberikan data masukan berupa proposisi a~, kemungkinan perbaikannya dengan menambahkan *operator* adalah:

```
====================
Menambahkan Operator
====================

Input anda:a~

Pilih operator logika:
   1. Konjungsi  : /\
      a/\~
   2. Disjungsi  : \/
      a\/~
   3. Implikasi  : =>
      a=>~
   4. Ekuvalensi : <=>
      a<=>~
Pilihan anda:_
```

Gambar 11. Perbaikan Operator Logika *Unary* Dengan Menambahkan Operator

Apabila diberikan data masukan berupa proposisi a/\~, kemungkinan perbaikannya adalah dengan menambahkan proposisi (Gambar 12).

```
====================
Memasukkan Proposisi
====================

Input anda:a/\~

Silahkan memasukkan proposisi:(a/\b)

Input anda yang telah diubah:a/\~(a/\b)
Apakah data masukan anda telah benar?(y/n)
_
```

Gambar 12, Perbaikan Operator Logika *Unary* Dengan Menambahkan Proposisi Baru

b) Operator logika *binary*.

Apabila diberikan sebuah data masukan berupa proposisi =>, kemungkinan perbaikannya adalah dengan menambahkan proposisi (Gambar 13).

```
====================
Memasukkan Proposisi
====================

Input anda:=>

Silahkan memasukkan proposisi:a

Input anda yang telah diubah:a=>
Apakah data masukan anda telah benar?(y/n)
_
```

Gambar 13. Perbaikan Pertama Operator Logika *Binary* Dengan Menambahkan Proposisi Baru

Karena perbaikan yang telah dilakukan belum sempurna, pengguna diminta untuk menambahkan proposisi kembali (Gambar 14).

```
====================
Memasukkan Proposisi
====================

Input anda:a=>

Silahkan memasukkan proposisi:b

Input anda yang telah diubah:a=>b
Apakah data masukan anda telah benar?(y/n)
_
```

Gambar 14. Perbaikan Kedua Operator Logika *Binary* Dengan Menambahkan Proposisi Baru

5. KESIMPULAN DAN SARAN

Pada makalah ini telah ditunjukkan bahwa pembuktian pernyataan logika proposisi dengan metode resolusi telah dapat dilakukan dengan bantuan piranti lunak yang dibuat. Piranti lunak dikembangkan dengan menggunakan bahasa pemrograman prosedural, C++. Pengujian yang telah dilakukan menunjukkan bahwa piranti lunak yang dibuat dapat melakukan pengubahan dan pembuktian dengan tingkat keberhasilan 100%. Pada piranti lunak tersebut terdapat fasilitas penanganan kesalahan yang dapat membantu pengguna dalam menggunakannya.

Pada pengembangan selanjutnya dapat dibuat piranti lunak untuk melakukan pembuktian pernyataan logika proposisi dengan metode lain, misalnya dengan memanfaatkan aturan ekivalensi pada aljabar boolean. Selain itu, dapat juga ditambahkan variasi penanganan kesalahan yang lebih banyak dengan melakukan studi mengenai efektifitas penggunaan piranti lunak oleh pengguna.

6. DAFTAR PUSTAKA

[1] Aribowo, A.; *Transformation of Propositional Logic Formula Into Conjunctive Normal Form With Prolog*, Tangerang, Indonesia : Jurnal Ilmiah Fakultas Ilmu Komputer, Universitas Pelita Harapan, Vol 3. No.3, 2005.

[2] Aribowo, A.; *Rancang Bangun Piranti Lunk Untuk Pengubahan Pernyataan Logika Proposisi Ke Dalam Bentuk Conjunctive Normal Form dan Pembuktian Dengan Metode Resolusi*, Tangerang, Indonesia : Jurnal Ilmiah Fakultas Ilmu Komputer, Universitas Pelita Harapan, Vol 6. No.2, 2008.

[3] Chang, C. L. dan Lee, R.C. *Symbolic Logic and Mechanical Theorem Proving*. 1975.

[4] Rosen, K.H., *Discrete Mathematics and Its Applications,* Fifth Edition, New York, USA : McGraw-Hill International Edition, 2003.

[5] Simpson, A., *Discrete Mathematics by Example*, UK : McGraw-Hill International Edition, 2002.