# *Aplikasi Konversi Ekuivalensi Logis Formula Proposisi dengan Pohon Biner*

Taufiq Hidayat[1], Muh. Nizomuddin Fauza Sidiq[2]
Jurusan Informatika Fakultas Teknologi Industri
Universitas Islam Indonesia
Yogyakarta
[1]taufiq.hidayat@uii.ac.id, [2]16523071@students.uii.ac.id

***Abstrak*— Makalah ini mempresentasikan hasil penelitian tentang pembuatan aplikasi yang dapat melakukan konversi formula proposisi ke formula lain yang ekuivalen secara logis. Konversi dilakukan dengan menggunakan hukum-hukum ekuivalensi dan dilakukan secara bertahap. Setiap tahap, aplikasi akan menentukan hukum-hukum yang bisa diterapkan pada formula yang diperoleh pada tahapan ini serta subformula yang akan dikenai hukum tersebut. Selain itu, aplikasi ini juga dapat mengecek apakah sebuah formula sudah dalam bentuk CNF. Dalam implementasinya, aplikasi ini menggunakan tipe data abstrak pohon biner untuk merepresentasikan sebuah formula proposisi.**

***Kata kunci—ekuivalensi logis, logika proposisi, CNF*, pohon.**

## I. PENDAHULUAN

Salah satu materi di Logika Matematika adalah mengubah sebuah formula menjadi formula lain yang ekuivalen secara logis. Dua formula disebut ekuivalen jika kedua formula mempunyai nilai kebenaran yang sama untuk setiap kombinasi nilai kebenaran variabel-variabel pembentuknya [1][2][3]. Untuk logika proposisi, salah satu logika yang dipelajari di Logika Matematik, terdapat 2 nilai kebenaran, yaitu benar (*true*) atau salah (*false*). Untuk selanjutnya simbol T untuk menyatakan nilai benar dan F untuk menyatakan nilai salah.

Contoh berikut ini adalah dua formula logika proposisi yang ekuivalen:

$$(p \wedge q) \rightarrow r$$

$$p \rightarrow (q \rightarrow r)$$

Kedua formula dibentuk dari 3 variabel proposisi, yaitu p, q, dan r. Karena keduanya mempunyai nilai kebenaran yang sama untuk setiap kombinasi nilai kebenaran p, q, dan r, kedua formula disebut ekuivalen.

Terdapat beberapa hukum untuk mengubah sebuah formula menjadi formula lain yang ekuivalen. Dengan hukum ekuivalensi, sebuah formula dapat diubah menjadi formula lain yang lebih sederhana. Hukum ekuivalensi diterapkan secara bertahap. Setiap tahapan hanya menggunakan satu hukum dan diterapkan terhadap salah satu subformula.[1][2]

Selain untuk menyederhanakan formula, keuntungan pengubahan formula dengan hukum-hukum ekuivalensi adalah dapat mengubah sebuah formula ke dalam bentuk normal (*normal form*), yang salah satunya adalah *conjunctive normal form (CNF)*. Formula dalam CNF menjadi obyek penelitian dalam *SAT problem*[2], yaitu problem apakah sebuah formula *satisfiable* atau tidak, dan *SAT solver*[2][4], yaitu perangkat lunak untuk menyelesaikan *SAT problem*. Saat ini *SAT solver* digunakan untuk menyelesaikan *SAT Problem* untuk formula dalam bentuk CNF. Di bidang Logika Matematika terapan, *SAT problem* dan *SAT solver* adalah subbidang yang paling banyak digunakan untuk menyelesaikan masalah-masalah nyata[4][5][6][7]. Selain terapan, penelitian di bidang ini juga berkaitan dengan algoritma untuk *SAT solver*[8][9][10][11].

Penggunaan hukum-hukum ekuivalensi memerlukan kejelian tersendiri. Sebelum melakukan konversi menjadi formula lain, perlu dikenali terlebih dahulu hukum-hukum yang bisa diterapkan karena tidak setiap hukum bisa diterapkan. Pengenalan ini didasarkan pada subformula-subformula di dalam formula tersebut.

Langkah berikutnya adalah memilih di antara hukum-hukum tersebut. Pemilihan ini didasarkan pada bentuk formula tujuan. Adakalanya pemilihan ini salah dan tidak mengarahkan ke formula tujuan. Langkah yang harus dilakukan adalah kembali ke tahapan sebelumnya. Tujuannya adalah memperbaiki langkah sebelumnya dengan memilih hukum ekuivalensi yang lain.

Berdasarkan masalah tersebut, penelitian ini akan membuat aplikasi yang dapat membantu melakukan konversi dari satu formula ke formula lain yang ekuivalen dengan menggunakan hukum-hukum ekuivalensi. Fitur yang harus tersedia di aplikasi:

1. mengenali hukum-hukum ekuivalensi yang bisa diterapkan di antara hukum-hukum yang ada dan menunjukkan subformula yang bisa dikenai hukum-hukum tersebut
2. mengembalikan ke formula sebelumnya jika pada sebuah tahapan tidak mengarahkan ke formula tujuan.
3. menentukan apakah sebuah formula sudah dalam CNF.

[a]Penelitian ini disponsori oleh Jurusan Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia dalam Program Hibah Kolaborasi Dosen dan Mahasiswa.

TABEL I. EKUIVALENSI DENGAN TABEL KEBENARAN

| $p$ | $q$ | $\neg p$ | $p \rightarrow q$ | $\neg p \vee q$ |
|---|---|---|---|---|
| T | T | F | T | T |
| T | F | F | F | F |
| F | T | T | T | T |
| F | F | T | T | T |

## II. LANDASAN TEORI

### A. Logika Proposisi dan CNF

Logika proposisi adalah logika yang didasarkan pada proposisi. Proposisi adalah sebuah pernyataan yang bisa ditentukan nilai kebenarannya [1][2][3]. Dapat ditentukan nilai kebenarannya berarti pernyataan tersebut hanya bisa bernilai salah satu dari nilai benar atau salah. Sebuah proposisi disimbolkan dengan huruf kecil yang menyatakan variabel proposisi Indeks bisa ditambahkan ke simbol huruf kecil tersebut jika diperlukan. Nilai benar dan salah disimbolkan dengan T dan F.

Sebuah variabel proposisi sudah membentuk sebuah formula proposisi. Untuk selanjutnya, formula proposisi akan disebut dengan formula saja . Formula yang lebih kompleks dapat dibentuk dengan menggabungkan beberapa formula dengan penghubung logika (operator logika), yaitu $\wedge$ (konjungsi), $\vee$ (disjungsi), $\rightarrow$ (implikasi), $\leftrightarrow$ (implikasi dua arah), dan $\neg$ (negasi). Selain negasi yang bersifat tunggal, operator lain bersifat biner.

Salah satu bentuk khusus dari formula proposisi adalah bentuk normal. Syarat formula dalam bentuk normal adalah [1][2][3]

1. hanya mengandung 3 operator alami dalam logika, yaitu konjungsi, disjungsi, dan negasi.
2. operator negasi hanya diterapkan terhadap variabel proposisi.

Jika semua operator disjungsi adalah subformula dari konjungsi dan tidak sebaliknya maka formula tersebut dalam CNF. Berikut ini adalah contoh formula dalam CNF:

1. $(p \vee q \vee \neg r) \wedge (\neg p \vee r)$
2. $p \wedge \neg q \wedge (q \vee r)$

### B. Ekuivalensi Logis dan Hukum Ekuivalensi

Secara informal, 2 formula disebut ekuivalen secara logis, yang selanjutnya disebut dengan ekuivalen, jika keduanya mempunyai makna yang sama. Sedangkan secara formal, 2 formula disebut ekuivalen jika kedua formula memberikan nilai kebenaran yang sama untuk setiap kombinasi nilai kebenaran dari setiap variabel pembentuknya.

Ekuivalensi 2 formula yang sederhana dan mengandung sedikit variabel bisa dibuktikan dengan menggunakan tabel kebenaran. Tabel kebenaran adalah tabel yang digunakan untuk menunjukkan nilai kebenaran formula untuk setiap kombinasi nilai variabel pembentuk. Tabel I adalah contoh pembuktian ekuivalensi formula: $p \rightarrow q$ dan $\neg p \vee q$. Dua kolom terakhir menunjukkan bahwa kedua formula memiliki nilai kebenaran yang sama. Dengan demikian kedua formula adalah ekuivalen.

Ekuivalensi dapat digunakan untuk menyatakan formula ke dalam bentuk lain yang lebih sederhana atau ke dalam bentuk CNF. Terdapat beberapa hukum ekuivalensi yang dapat digunakan untuk mengubah formula ke dalam bentuk lain yang ekuivalen. Meskipun ada perbedaan antar referensi, secara umum ada 15 hukum ekuivalensi, yaitu[1][3]:

1. Komutatif
   $A \wedge B \equiv B \wedge A$
   $A \vee B \equiv B \vee A$
   $A \leftrightarrow B \equiv B \leftrightarrow A$
2. Asosiatif
   $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$
   $(A \vee B) \vee C \equiv A \vee (B \vee C)$
   $(A \leftrightarrow B) \leftrightarrow C \equiv A \leftrightarrow (B \leftrightarrow C)$
3. Distributif
   $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$
   $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$
4. Identitas
   $A \wedge T \equiv A$
   $A \vee F \equiv A$
5. *Zero*
   $A \wedge F \equiv F$
   $A \vee T \equiv T$
6. De Morgan
   $\neg(A \wedge B) \equiv \neg A \vee \neg B$
   $\neg(A \vee B) \equiv \neg A \wedge \neg B$
7. *Idempotent*
   $A \wedge A \equiv A$
   $A \vee A \equiv A$
8. Negasi Ganda
   $\neg\neg A \equiv A$
9. Tautologi
   $A \vee \neg A \equiv T$
10. Kontradiksi
    $A \wedge \neg A \equiv F$
11. *Absorption* (Penyerapan)
    $A \wedge (A \vee B) \equiv A$
    $A \vee (A \wedge B) \equiv A$

$A \wedge (\neg A \vee B) \equiv A \vee B$

$A \vee (\neg A \wedge B) \equiv A \wedge B$

12. Kontraposisi

    $A \rightarrow B \equiv \neg B \rightarrow \neg A$

13. Ekspor

    $(A \wedge B) \rightarrow C \equiv A \rightarrow (B \rightarrow C)$

14. Ekuivalensi Implikasi

    $A \rightarrow B \equiv \neg A \vee B$

15. Ekuivalensi Implikasi Dua Arah

    $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$

A, B, dan C adalah formula sembarang, yang dapat berupa formula sederhana maupun formula yang kompleks. Hukum ekuivalensi bersifat 2 arah, dari kiri ke kanan atau dari kanan ke kiri. Hukum-hukum ini bisa diterapkan terhadap subformula.

### *C. Pohon dan Pohon Biner*

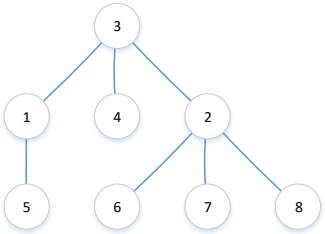
Pohon adalah tipe data abstrak yang tergolong tipe data abstrak nonlinier. Sebuah pohon terdiri dari simpul-simpul dan sisi-sisi yang menghubungkan antarsimpul. Terdapat sebuah simpul khusus yang diberi nama akar (*root*), yang diilustrasikan terletak pada posisi paling atas. Beberapa simpul yang terhubung langsung dengan sisi ke akar diletakkan di bawahnya. Simpul-simpul lain yang terhubung dengan simpul-simpul tersebut diletakkan di bawahnya, demikian seterusnya. Simpul-simpul yang tidak mempunyai sisi ke simpul di bawahnya disebut sebagai daun (*leaf*)[12][13]. Ciri-ciri pohon yang membedakan dengan tipe data abstrak non-linier lain, yaitu graf, adalah bahwa hanya ada 1 jalur yang menghubungkan satu simpul ke simpul yang lain. Gambar 1 adalah contoh sebuah pohon. Simpul 3 disebut akar, sedangkan simpul 5, 6, 7, dan 8 disebut daun.

Pohon biner adalah pohon dalam bentuk khusus. Setiap simpul pada pohon biner terhubung dengan simpul di bawahnya maksimal sebanyak 2 simpul.

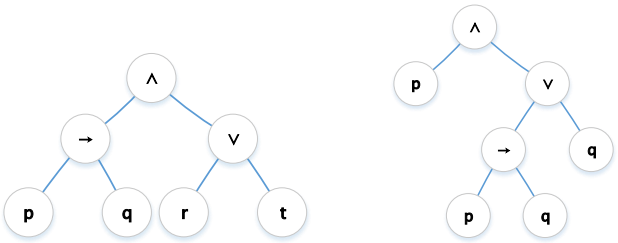
## III. METODOLOGI

Sesuai dengan uraian pada Bab I, aplikasi konversi formula ke formula lain yang ekuivalen harus memiliki fitur-fitur sebagai berikut:

1. menunjukkan proses konversi
2. menentukan hukum-hukum ekuivalensi yang bisa diterapkan terhadap formula pada satu tahapan proses konversi




Gambar 1. Contoh pohon




Gambar 2. Formula dalam pohon biner

3. menentukan subformula-subformula yang bisa dikonversi sesuai dengan hukum-hukum ekuialensi yang bisa diterapkan pada fitur 2
4. melakukan pengecekan apakah sebuah formula dalam kondisi CNF, dan
5. melakukan pembatalan sebuah tahapan konversi dan kembali ke tahapan sebelumnya.

Langkah- langkah penelitian yang akan diuraikan di bab ini adalah sebagai berikut:

1. Desain: membahas tentang desain aplikasi yang akan dibuat, yang lebih menekankan pada desain struktur data.
2. Implementasi: membahas tentang pembuatan aplikasi terutama tentang tampilan
3. Pengujian: membahas tentang proses dan hasil pengujian.
4. Analisis Metode: membahas tentang perbandingan dengan tipe data abstrak selain pohon.

### *A. Desain*

Pembahasan dalam desain lebih ditekankan pada hal-hal yang berkaitan dengan fitur-fitur yang ada dalam aplikasi. Berkaitan dengan hal ini, salah satu bagian desain yang sangat penting adalah pemilihan struktur data. Pemilihan struktur data yang tepat akan memudahkan pembuatan proses-proses yang diperlukan dalam fitur-fitur yang harus tersedia dalam aplikasi. Untuk desain ini, fitur 2, 3, dan 4 adalah fitur yang harus dipertimbangkan.

#### *1) Representasi Formula dengan Pohon Biner*

Tipe data abstrak pohon biner adalah tipe data yang cocok untuk merepresentasikan ekspresi matematika. Simpul pohon biner menyimpan informasi tentang operator matematika, cabang kiri untuk operan yang berada di sisi kiri operator sedangkan cabang kanan untuk operan yang berada di sisi kanan. Apabila operator bersifat tunggal, cukup menggunakan cabang kiri untuk menyimpan operan. Demikian juga dengan formula di Logika Matematika, dapat direpresentasikan dengan pohon biner karena operator di Logika Matematika hanya memiliki 2 kemungkinan, yaitu tunggal atau biner.

Gambar 2 adalah contoh representasi formula dengan pohon. Gambar sebelah kiri untuk formula $(p \rightarrow q) \wedge (r \vee t)$, sedangkan gambar sebelah kanan untuk formula $p \wedge ((p \rightarrow q) \vee q)$.

#### *2) Pemilihan Hukum Ekuivalensi*

Untuk menentukan hukum ekuivalensi yang bisa diterapkan terhadap sebuah formula, dapat dikenali dari pola Pohon formula tersebut. Pola yang harus dikenali juga dilakukan terhadap subpohon karena pengubahan formula bisa dilakukan terhadap subformula. Dengan demikian, perlu ditelusuri setiap subpohon dalam pemilihan hukum ekuivalensi ini.

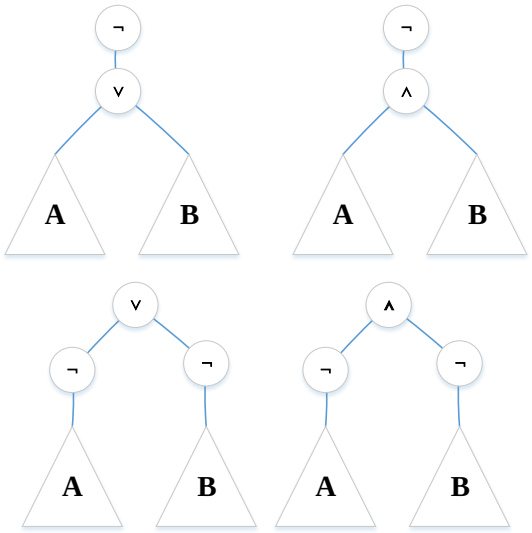
Sebagai contoh adalah hukum De Morgan. Ada 4 pola subpohon yang perlu dikenali. Dua pola untuk mengubah dari formula sisi kiri ke kanan dan 2 pola untuk mengubah dari formula sisi kanan ke sisi kiri. Keempat pola tersebut dapat dilihat pada Gambar 3.

*3) Konversi* Pohon

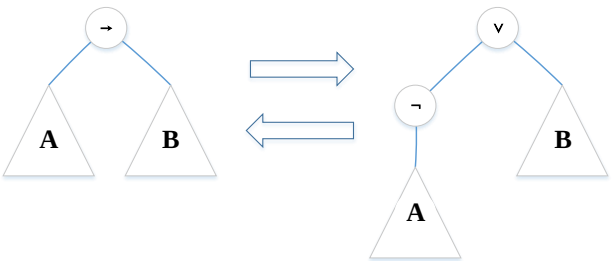
Konversi formula ke bentuk formula lain dilakukan dengan melakukan konversi pohon. Konversi dilakukan sesuai dengan hukum yang diterapkan dan subpohon yang akan dikenai. Dengan pohon, konversi akan mudah dilakukan, yaitu mengganti pola awal ke pola tujuan. Jika konversi menggunakan hukum ekuivalensi dari kiri kanan maka pola awal adalah pola subpohon untuk sisi kiri dan pola tujuan adalah pola subpohon untuk sisi kanan. Hal ini berlaku juga untuk arah kebalikannya.

Gambar 4 adalah contoh konversi pohon untuk hukum ekuivalensi implikasi. Gambar pohon yang di sebelah kiri adalah pola untuk formula sisi kiri hukum tersebut sedangkan pohon sebelah kanan adalah pola formula untuk sisi kanan.

*4) Pengecekan CNF*




Gambar 3. Pola subpohon untuk Hukum De Morgan




Gambar 4. Konversi pohon untuk hukum ekuivalensi implikasi

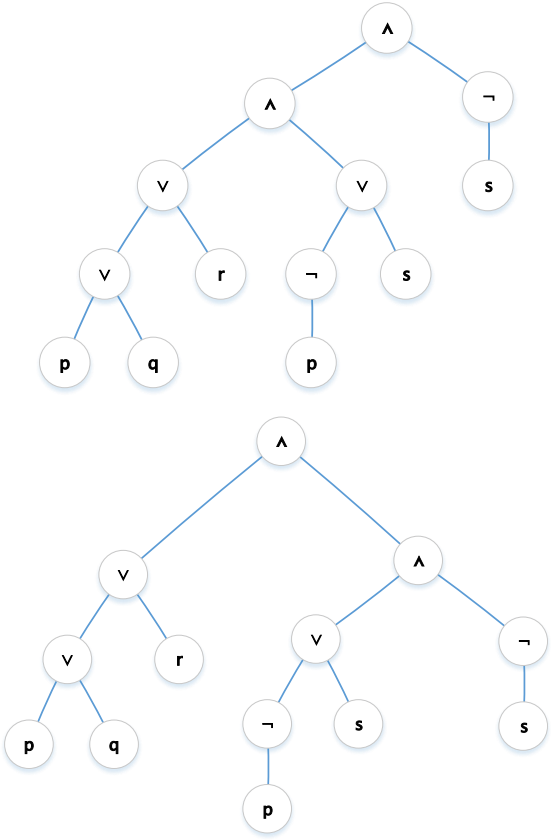
Salah satu fitur yang disediakan di aplikasi ini adalah kemampuan untuk mengecek apakah formula pada sebuah tahapan sudah termasuk CNF atau belum. Pengecekan ini sedikit bermasalah karena ada perbedaan notasi di dalam representasi di pohon biner . Biasanya CNF dinyatakan dalam bentuk rangkaian konjungsi terhadap subformula yang disebut sebagai klausa, setiap klausa dinyatakan dalam bentuk rangkaian disjungsi dari *literal*, dan setiap *literal* berupa variabel proposisi atau negasi dari variabel proposisi. Contoh formula CNF adalah sebagai berikut:

$$(p \vee q \vee r) \wedge (\neg p \vee s) \wedge \neg s$$

Karena formula direpresentasikan dengan menggunakan pohon biner, pohon untuk formula CNF tersebut bisa dalam berbagai bentuk yang berbeda meskipun maknanya sama. Gambar 5 adalah beberapa kemungkinan bentuk pohon untuk formula CNF tersebut.

Meskipun bentuknya berbeda tetapi setiap pohon mempunyai sifat yang sama terkait dengan formula CNF. Sifat tersebut adalah setiap jalur dari akar ke setiap daun memenuhi ciri-ciri yaitu: urutan simpul pada setiap jalur adalah rangkaian simpul konjungsi jika ada, rangkaian simpul disjungsi jika ada, 1 simpul negasi atau tidak ada, dan simpul variabel proposisi. Dengan kata lain, sebuah pohon adalah representasi formula CNF jika memenuhi kriteria berikut:

1. setiap simpul konjungsi terhubung di bawahnya dengan simpul konjungsi lain, simpul disjungsi, simpul negasi, atau simpul variabel,




Gambar 5. Representasi Pohon yang berbeda untuk formula yang sama.

2. setiap simpul disjungsi terhubung di bawahnya dengan simpul disjungsi lain, simpul negasi, atau simpul variabel, dan
3. setiap simpul negasi hanya terhubung di bawahnya dengan simpul variabel.

## B. *Implementasi*

Aplikasi dibuat dengan menggunakan bahasa Java[13] dan berbasis GUI. Gambar 6 adalah tampilan utama aplikasi yang dibangun. Beberapa bagian penting, yang merupakan fitur utama aplikasi, ditandai dengan nomor 1 sampai dengan nomor 4. Penjelasan bagian per nomor adalah sebagai berikut:

1. Daftar formula (fitur 1 dan fitur 5)

   Menampilkan proses konversi. Setiap baris menyatakan formula pada sebuah tahapan disertai dengan keterangan tentang hukum ekuivalensi yang diterapkan untuk mendapatkan formula tersebut.

   Selain menampilkan formula, daftar ini juga menyimpan formula setiap tahapan. Dengan demikian, fitur untuk membatalkan satu tahapan dan kembali ke tahapan sebelumnya dapat dilakukan. Tombol *Undo* digunakan untuk tujuan ini.

2. Hukum-hukum ekuivalensi yang bisa diterapkan (fitur 2)

   Menampilkan semua hukum-hukum ekuivalensi. Hukum ekuivalensi yang bisa diterapkan pada formula di satu tahapan (tahapan terakhir) ditandai sebagai aktif, sedangkan yang tidak bisa diterapkan ditandai sebagai non-aktif.

3. Subformula (fitur 3)

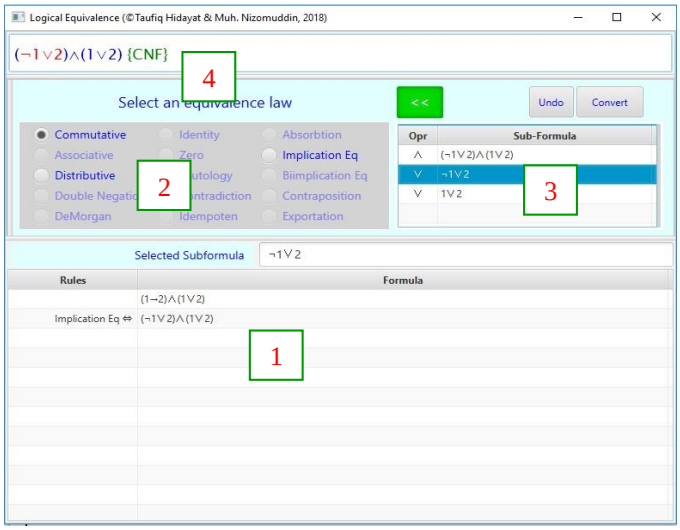
   Menampilkan kemungkinan semua subformula yang bisa dikonversi apabila salah satu hukum ekuivalensi telah dipilih.

4. Formula dan penekanan subformula (fitur 3 dan fitur 5)

   Menampilkan formula pada sebuah tahapan. Subformula yang dipilih di bagian subformula akan diberi warna yang berbeda untuk menunjukkan posisi subformula yang akan dikonversi.

   Informasi tulisan CNF akan ditambahkan jika formula sudah dalam bentuk CNF.

Dalam aplikasi ini, variabel dinyatakan dalam bentuk angka dengan tujuan agar cacah variabel yang dipakai bisa banyak.




Gambar 6. Tampilan aplikasi Konversi formula proposisi.

## C. *Pengujian*

### 1) *Pengujian Fitur*

Pengujian ini bertujuan untuk membuktikan bahwa fitur-fitur yang diperlukan sudah berjalan dengan benar. Aplikasi akan diuji dengan formula $1 \leftrightarrow 2$. Gambar 7 adalah tampilan setelah 2 kali dilakukan konversi dengan hukum ekuivalensi implikasi dua arah untuk $1 \leftrightarrow 2$ dan ekuivalensi implikasi untuk $2 \rightarrow 1$. Pada tahapan ini, diperoleh informasi sebagai berikut:

1. Proses konversi:

   Daftar formula menampilkan proses konversi setelah penerapan hukum ekuivalensi implikasi dua arah dan ekuivalensi implikasi sehingga diperoleh

   $1 \leftrightarrow 2$

   $\equiv (1 \rightarrow 2) \wedge (2 \rightarrow 1)$ {ekuivalensi implikasi 2 arah}

   $\equiv (1 \rightarrow 2) \wedge (\neg 2 \vee 1)$ {ekuivalensi implikasi}

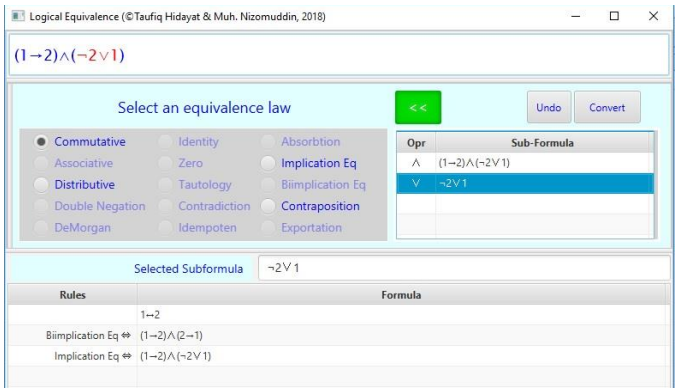
   Berdasarkan hukum-hukum ekuivalensi di Bab 0, hasil yang ditampilkan sudah benar.

2. hukum ekuivalensi yang bisa diterapkan

   Formula setelah 2 tahapan adalah $(1 \rightarrow 2) \wedge (\neg 2 \vee 1)$. Berdasarkan hukum-hukum ekuivalensi yang ada, yang bisa diterapkan terhadap formula ini adalah hukum komutatif, distributif, ekuivalensi implikasi, dan kontraposisi. Gambar 7 menunjukkan keempat hukum tersebut aktif sedangkan yang lain non-aktif.

3. subformula sesuai hukum ekuivalensi yang bisa diterapkan

   Gambar 7 juga menunjukkan bahwa ada 2 subformula yang bisa dikonversi jika dipilih hukum komutatif, yaitu untuk operator konjungsi dan operator disjungsi.




Gambar 7. Pengujian proses konversi

Gambar 10. Formula logika proposisi dengan senarai.

Jika yang dipilih adalah operator disjungsi maka subformula $\neg 2 \vee 1$ pada formula akan berwarna merah.

4. kondisi CNF

   Formula pada tahapan ini tidak dalam kondisi CNF karena memuat operator implikasi. Formula pada Gambar 7 tidak menampilkan informasi CNF.

5. pembatalan tahapan proses konversi

   Tombol *Undo* pada_Gambar 7 dalam kondisi aktif untuk menunjukkan bahwa pembatalan konversi dapat dilakukan.

*2) Pengujian Aplikasi*

Pengujian ini bertujuan untuk membuktikan bahwa hasil konversi akhir adalah formula yang ekuivalen dengan formula masukan. Aplikasi diuji dengan metode *Black Box* untuk 2 kasus, yaitu konversi formula menjadi bentuk yang sederhana dan konversi formula menjadi bentuk CNF. Kasus pertama akan dibuktikan bahwa $q \leftrightarrow (q \vee p) \equiv p \rightarrow q$ yang sudah dibuktikan pada halaman 36-37 di buku [3] sedangkan kasus kedua adalah mengubah $p \leftrightarrow (q \wedge r)$ ke formula CNF $(\neg p \vee q) \wedge (\neg p \vee r) \wedge (\neg q \vee \neg r \vee \neg p)$ yang sudah dibuktikan pada halaman 49 dalam [3] .

Gambar 8 menunjukkan keluaran aplikasi untuk kasus 1 sedangkan Gambar 9 adalah keluaran aplikasi untuk kasus 2. Keluaran yang ditampilkan menunjukkan hasil yang sama dengan formula tujuan sehingga bisa dibuktikan bahwa hasil akhir yang diperoleh adalah formula yang ekuivalen dengan formula masukan.

| Rules | Formula |
|---|---|
| | 2↔(2∨1) |
| Biimplication Eq ⇔ | (2→(2∨1))∧((2∨1)→2) |
| Implication Eq ⇔ | (¬2∨(2∨1))∧((2∨1)→2) |
| Implication Eq ⇔ | (¬2∨(2∨1))∧(¬(2∨1)∨2) |
| Associative ⇔ | ((¬2∨2)∨1)∧(¬(2∨1)∨2) |
| Commutative ⇔ | ((2∨¬2)∨1)∧(¬(2∨1)∨2) |
| Tautology ⇔ | (T∨1)∧(¬(2∨1)∨2) |
| Commutative ⇔ | (1∨T)∧(¬(2∨1)∨2) |
| Zero ⇔ | T∧(¬(2∨1)∨2) |
| Commutative ⇔ | (¬(2∨1)∨2)∧T |
| Identity ⇔ | ¬(2∨1)∨2 |
| Commutative ⇔ | 2∨¬(2∨1) |
| DeMorgan ⇔ | 2∨(¬2∧¬1) |
| Absorbtion ⇔ | 2∨¬1 |
| Commutative ⇔ | ¬1∨2 |
| Implication Eq ⇔ | ¬(¬1)→2 |
| Double Negation ⇔ | 1→2 |

Gambar 8. Pembuktian 2 formula yang ekuivalen.

| Rules | Formula |
|---|---|
| | 1↔(2∧3) |
| Biimplication Eq ⇔ | (1→(2∧3))∧((2∧3)→1) |
| Implication Eq ⇔ | (¬1∨(2∧3))∧((2∧3)→1) |
| Distributive ⇔ | ((¬1∨2)∧(¬1∨3))∧((2∧3)→1) |
| Implication Eq ⇔ | ((¬1∨2)∧(¬1∨3))∧(¬(2∧3)∨1) |
| DeMorgan ⇔ | ((¬1∨2)∧(¬1∨3))∧((¬2∨¬3)∨1) |

Gambar 9. Pengubahan formula menjadi formula CNF.

*3) Pengujian oleh Pengguna*

Aplikasi ini diujikan ke 10 mahasiswa Informatika, Universitas Islam Indonesia, yang sedang mengambil kuliah Logika Matematika pada Semester Ganjil tahun akademik 2017/2018 dan menjawab kuesioner. Dari 10 mahasiswa, 1 orang menyatakan bahwa tidak yakin sudah mempelajari Logika Matematika sehingga jawaban dari responden ini dikeluarkan dari analisis karena aplikasi ini tidak ditujukan untuk pengguna yang belum pernah belajar.

Sebanyak 78% responden menyatakan bahwa aplikasi ini dapat membantu belajar ekuivalensi. Responden ini adalah responden yang menganggap bahwa materi ekuivalensi adalah sulit, mudah, atau biasa saja jika dibandingkan dengan materi-materi lain di Logika Matematika. Dari responden yang menganggap materi ekuivalensi sulit, 75% menyatakan bahwa aplikasi ini dapat membantu mereka sendiri dalam belajar ekuivalensi, sisanya menyatakan tidak tahu. Responden yang menjawab tidak tahu menyatakan bahwa mereka menggunakan huruf sebagai variabel saat belajar Logika Matematika.

Komentar yang banyak disampaikan oleh responden berkaitan dengan penggunaan angka sebagai variabel. Sebanyak 44% responden menyarankan penggunaan huruf.

## *D. Analisis Metode*

Aplikasi ini menggunakan pohon biner untuk merepresentasikan formula logika proposisi. Tipe data abstrak lain yang kemungkinan bisa digunakan adalah senarai (*list*). Tipe data abstrak lain seperti tumpukan (*stack*), antrian (*queue*), dan graf dapat diabaikan untuk representasi formula. Tumpukan dan antrian sama dengan senarai dalam hal representasi. Yang membedakan ketiganya hanya dalam hal operasi. Sedangkan graf adalah tipe data yang lebih umum dibandingkan dengan pohon.

Senarai ini lebih sederhana dibandingkan Pohon. Gambar 10 adalah contoh representasi formula logika proposisi dengan senarai untuk formula $p \wedge ((p \rightarrow q) \vee q)$.

Namun, dengan senarai, fitur-fitur yang penting di aplikasi lebih sulit diimplementasikan. Contoh fitur tersebut adalah penentuan hukum ekuivalensi yang bisa diterapkan karena harus mencocokkan pola formula dengan pola hukum ekuivalensi.

Sebagai contoh adalah salah satu hukum yang sederhana yaitu komutatif, yang berlaku terhadap operator konjungsi. Hukum ini bisa diterapkan terhadap formula pada Gambar 10 tetapi pencocokan pola ini tidak sederhana. Aplikasi harus menentukan bahwa konjungsi ini berlaku untuk $p$ dan $(p \rightarrow q) \vee q$, dan bukan untuk $p$ dan $p \rightarrow q$ . Proses ini lebih sulit dibandingkan dengan proses yang sama untuk representasi pohon biner, yaitu dengan hanya mengambil simpul kiri dan simpul kanan dari simpul konjungsi. Representasi formula tersebut dengan pohon biner ditunjukkan pada gambar sebelah kanan dari Gambar 2.

Demikian juga pada saat dilakukan konversi. Konversi senarai pada Gambar 10 dengan hukum komutatif untuk

operator konjungsi lebih sulit dilakukan. Konversi pohon biner pada Gambar 2 untuk hukum yang sama lebih mudah, yaitu dengan hanya melalui pertukaran simpul kiri dan simpul kanan saja.

Masalah ini berlaku juga untuk hukum-hukum yang lain, yang sebagian besar lebih rumit dibandingkan dengan hukum komutatif. Dengan demikian dapat disimpulkan bahwa pohon biner lebih memudahkan penentuan hukum ekuivalensi yang bisa diterapkan dan proses konversi.

## IV. KESIMPULAN

Telah berhasil dibangun aplikasi yang mampu melakukan konversi formula proposisi ke bentuk lain yang ekuivalen secara logis, Aplikasi ini menggunakan tipe data abstrak pohon, khususnya pohon biner untuk merepresentasikan formula. Representasi pohon ini memudahkan penentuan hukum-hukum ekuivalensi yang bisa diterapkan terhadap formula dan memudahkan dalam konversi formula dengan menggunakan sebuah hukum ekuivalensi.

## *Referensi*


[1] S. Hedman, "A First Course in Logic", New York: Oxford University Press, 2006.

[2] M. Huth dan M. Ryan, "Logic in Computer Science: Modelling and Reasoning about Systems", New York: Cambridge University Press, 2004.

[3] T. Hidayat, "Logika Proposisi", Yogyakarta: Dar Fiqrin, 2014.

[4] Y. Vizel, G. Weissenbacher, dan S. Malik, "Boolean Satisfiability Solvers and Their Applications in Model Checking", Proceedings of the IEEE., 103, 2015.

[5] I. Lynce, dan J. Ouaknine, "Sudoku as a SAT problem", in The Proceedings of AIMATH'06, 2006.

[6] T. Hidayat, "Attribute exploration of many-valued context with SAT", Proceedings of 10th WSEAS international conference on Computational Intelligence, Man-Machine Systems and Cybernetics, pp. 33-37, 2011.

[7] T. Hidayat, "Using Sat for Attribute Exploration of Formal Context with Constraint", Proceedings of International Conference on Information Technology 2013, 2013.

[8] C. M. Li, dan Anbulagan, "Heuristics Based on Unit Propagation for Satisfiability Problems", Proceedings of IJCAI, 366-371, 1997.

[9] U. Schoning, "A probabilistic algorithm for k-SAT and constraint satisfaction problems", Proceedings of 40th Annual Symposium on Foundations of Computer Science, pp. 410–414, 1999.

[10] J. P. Marques-Silva, dan K. A. Sakallah, "GRASP: A New Search Algorithm for Satisfiability", Proceedings of International Conference on Computer-Aided Design, pp. 220-227, 1996.

[11] M. Moskewicz, et.al. "Chaff: Engineering an Efficient SAT Solver", 39th Design Automation Conference (DAC 2001), ACM 2001.

[12] A. Gopal, "Magnifying Data Structures", PHI Learning, 2010.

[13] M. A. Weiss, "Data Structures and Algorithm Analysis in Java", 3rd edition, Pearson, 2011.