BAB II
TINJAUAN PUSTAKA

Pada tinjauan pustaka atau kajian literatur ini berisi studi pustaka terhadap buku, jurnal ilmiah, maupun penelitian sebelumnya yang berkaitan dengan tulisan ini antara lain SAT-SOLVER, DPLL *(Davis-Putnam-Logemann-Loveland)*, dan CNF (*conjunctive normal form)*-SAT Problem. Uraian tinjauan pustaka dimaksudkan untuk menyusun kerangka pemikiran atau konsep yang akan digunakan dalam menyelesaikan Tugas Akhir ini. Adapun Kajian Literatur ini memuat kajian literatur induktif dan deduktif.

2.1 Kajian Literatur Induktif

Kajian literatur *induktif* adalah kajian yang bersifat khusus yang diperoleh dari jurnal, karya ilmiah, Koran dan sebagainya.Kajian ini berpedoman pada pengamatan dahulu atau pengamatan sebelumnya, kemudian berdasarkan pengamatan tersebutbarulah di tarik kesimpulan.

Untuk Kajian Literatur *Induktif*saat ini sudah ada *paper* mengenai penelitian SAT Solver dan metode DPLL *(Davis-Putnam-Logemann-Loveland)* oleh Taufiq Hidayat & Agung Bahariyanto I, pada tahun 2018 dalam makalah Seminar Nasional Aplikasi Teknologi Informasi (SNATI) mereka menulis tentang *"SAT Solver dengan DPLL dalam pemograman deklaratif"*. Penelitian ini tentang bagaimana pembuatan SAT Solver menggunakan pemrograman deklaratif dan juga bahasa Prolog serta menggunakan Algoritma *Davis-Putnam-Logemann-Loveland*(DPLL). Disini Taufiq dan Agung menyatakan bahwa penelitiannya ini merupakan bagian dari penelitian tentang sistem *eksplorasi Formal Context* dengan *constraint*. Lebih lanjut Taufiq mengatakan SAT Solver ini lebih menekankan pada analisa logika, bukan matematika yang selama ini banyak berkembang. Saat ini, SAT Solver membutuhkan waktu 30 detik, sedangkan pemecahan dengan program yang berdasarkan matematika hanya satu detik, namun ke depan, kata Taufiq, SAT Solver ini akan lebih cepat karena pemecahannya dibantu komputer. Sedangkan tingkat ketepatan pemecahan masalah, SAT Solver lebih unggul, dimana setelah pengujian akhirnya diketahui bahwa SAT Solver dalam penelitian ini mampu menyelesaikan SAT Problem. Salah satu problem yang diujikan oleh Taufiq dan Agung adalah problem Sudoku, dimana SAT Problem dengan 729 variabel dan lebih dari 8829 klausa. (Garuda Ristek Dikti ,2018)

Taufiq Hidayat & Muh. Nizomuddin FS (2018) juga pernah meneliti mengenai pembuatan perangkat lunak yang dapat mengkonversi formula proposisi ke formula lain namun tetap memiliki nilai ekuivalen yang logis yang pada penelitian ini konversi dilakukan menggunakan hukum-hukum ekuivalensi dan dilakukan secara bertahap. Setiap tahap, aplikasi akan mencari dan menentukan aturan / hukum yang bisa diterapkan pada formula yang diperoleh pada tahapan tersebut serta subformula yang akan dikenai hukum tersebut. Selain itu, aplikasi tersebut juga dapat melakukan pengecekan bentuk sebuah formula apakah formula tersebut sudah dalam bentuk CNF atau belum.Dalam implementasiannya, aplikasi tersebut menggunakan tipe data bentuk abstrak pohon biner untuk merepresentasikan sebuah formula proposisi. Berdasarkan hasil penelitian tersebut maka berhasillah dibangunnya sebuah aplikasi yang mampu mengkonversi formula proposisi ke bentuk lain yang ekuivalen secara logis. Aplikasi tersebut menggunakan tipe data abstrak pohon, khususnya pohon biner untuk merepresentasikan formula yang diberikan. Representasi pohon ini memberikan kemudahan dalam menentukan bentuk ekuivalensi dan aturan atau hokum ekuivalensi yang bisa diterapkan terhadap formula yang akhirnya mempermudah dalam mengkonversikan formula dengan menggunakan sebuah hukum ekuivalensi.(Garuda Ristek Dikti ,2018)

Paul Jackson dan Daniel Sheridan pada tahun 2005 juga menulis tentang *"Clause Form Conversions For Boolean Circuits".* Penelitian ini tentang pengontrolan transmisi arus listrik dalam sirkuit dengan menggunakan CNF dapat mempersingkat waktu yang dibutuhkan.

2.2 Kajian Literatur Deduktif

Kajian deduktif adalah kajian yang diperoleh dari buku – buku atau literatur yang berisi tentang teori–teori, landasan teori yang berhubungan dengan penelitian yang dilakukan yang berbasiskan logika untuk mendapatkan satu atau lebih kesimpulan berdasarkan satu atau lebih premis yang diberikan. Dalam sistem deduktif yang kompleks, peneliti dapat mendapatkan lebih dari satu kesimpulan. Metode deduktif sering di ibaratkan seperti pengambilan kesimpulan dari sesuatu hal umum ke khusus.(Talentsecret,2013).

Untuk Kajian *Deduktif* di kutip lah tentang teori – teori dasar yang berkaitan dengan Pengertian Implementasi CNF SAT Solver dengan metode algoritme DPLL menggunakan AHK script. Beberapa teori dasar tersebut adalah :

a. SAT Problem (Boolean Satisfiability Problem)

Dalam ilmu komputer ,Boolean Satisfiability Problem (kadang-kadang disebut satisfisability problem dan disingkat sebagai SATISFIABILITY atau SAT Problem) adalah suatu problem/permasalahan untuk menentukan apakah sebuah kalimat dapat dipenuhi (satisfiable) atau tidak sesuai formula Boolean yang diberikan. Dengan kata lain, ia menanyakan apakah variabel-rumus rumus Boolean yang diberikan dapat secara konsisten digantikan oleh nilai-nilai TRUE atau FALSE sedemikian rupa sehingga rumus tersebut bernilai TRUE .Jika ini kasusnya, rumusnya disebut *satisfiable* atau memuaskan. Di sisi lain, jika tidak seperti itu, fungsi yang dinyatakan oleh rumus adalah FALSE untuk semua variabel dan rumusnya *tidak memuaskan* . Misalnya, rumus " *a* DAN TIDAK *b* " ini dianggap satisfiable atau memuaskan karena seseorang dapat menemukan nilai *a*sebagai TRUE dan *b*sebagai FALSE, yang membuat (*a* DAN NOT *b*) = TRUE. Sebaliknya, " α DAN BUKAN α " dianggap tidak memuaskan (unsatisfiable).

Adapun Rumus logika proposisi disebut ekspresi Boolean yang dibangun dari variabel , operator DAN (konjungsi , juga dilambangkan dengan ∧), ATAU (disjungsi , ∨), BUKAN (negasi , ¬), dan tanda kurung. Rumus dikatakan *satisfiable* jika dapat dibuat BENAR dengan menetapkan nilai logis yang sesuai (yaitu BENAR, SALAH) untuk variabel-variabelnya. *satisfiability* problem Boolean (SAT), diberikan rumus, untuk memeriksa apakah itu *satisfiable*. Masalah keputusan ini sangat penting di berbagai bidang ilmu komputer , termasuk teori ilmu komputer , teori kompleksitas , algoritmik , kriptografi dan kecerdasan buatan.

b. SAT Solver

SAT Solver merupakan suatu perangkat lunak (*Software)* yang memiliki fungsi untuk menyelesaikan sebuah SAT Problem *(satisfiability problem),* dimana perangkat lunak berfungsi untuk menentukan apakah sebuah formula logika proposisi itu memiliki nilai *satisfiable* atau tidak (*unsatisfiable).* Formula logika proposisi yang dapat diselesaikan dengan SAT Solver biasanya memiliki bentuk format CNF *(conjunctive normal form).* Dalam penelitian yang terdahulu, SAT Solver biasa digunakan dalam penyelesaian eksplorasi atribut *Formal Context* dengan batasan *(constraint).*

Langkah penyelesaian problem entailment dengan SAT Solver adalah sebagai berikut:

1. Problem entailment dinyatakan sebagai SAT *Problem*.

2. SAT *Problem* disimpan pada sebuah file teks.
3. SAT Solver dijalankan dengan *input* berupa file teks untuk *entailment problem*.(Jackson, 2005).

c. Algoritma DPLL (Davis-Putnam-Logemann-Loveland)

Sebelum membahas tentang DPLL secara detail, perlu diketahui terlebih dahulu tentang algoritma. Istilah algoritma digunakan dalam ilmu komputer untuk menggambarkan metode pemecahan masalah yang terbatas, deterministik, dan efektif yang cocok untuk implementasi sebagai program komputer (Sedgewick & Wayne, 2011).

Algoritma terletak di jantungnya komputer (Cormen et al, 2009). Dan sekarang ini sudah sangat banyak algoritma yang ditemukan seiring dengan berkembang pesatnya teknologi dan komputer yang merupakan salah satu sarana untuk memfasilitasi dan mengimplementasikan algoritma - algoritma tersebut. Menurut Shabanah (2010) memahami algoritma adalah salah satu aspek yang paling sulit dari studi Ilmu Komputer karena algoritma biasanya memodelkan konsep yang rumit, merujuk pada gagasan matematika yang abstrak, atau menggambarkan perubahan dinamis yang kompleks dalam struktur data untuk menyelesaikan masalah yang relatif sulit.

Secara sederhana algoritma adalah urutan langkah langkah yang logis untuk menyelesaikan suatu masalah. Urutan langkah yang logis berarti algoritma harus ada suatu standar atau metode dengan suatu urutan tertentu secara berurutan, tidak boleh melompat-lompat.Secara definisi, algoritma merupakan alur pemikiran seseorang yang logis, yang dapat diterima orang lain dan dituangkan dalam bentuk tulisan. Karena yang menjadi ukuran adalah alur pikiran seseorang, maka algoritma bagi seseorang bisa tidak sama dengan algoritma orang yang lain. Karena algoritma juga merupakan sesuatu yang tertulis, maka algoritma dapat berupa kalimat, gambar, atau tabel tertentu.

Algoritma *Davis-Putnam-Logemann-Loveland*(DPLL) adalah suatu algoritma yang dikembangkan oleh Martin Davis, Hilary Putnam, George Logemann, dan Donald Loveland, pada tahun 1960-an untuk dapat menyelesaikan masalah SAT Problem dalam bentuk *Conjunctive Normal Form* (CNF). Walaupun algoritma ini sudah sangat lama dikembangkan, namun sampai saat ini masih banyak pemecah SAT (*SAT Solver*) yang dibuat berdasarkan algoritma ini. Algoritma ini merupakan algoritma yang dibuat berdasarkan pengembangan dari Algoritma DPLL.Algoritma ini berbasis pada 5 aturanyaitu *unit propagation, pure literal, decide, fail (unsatisfiable),* dan *backtrack.* Aturan-aturan ini akan

diterapkan terhadap formula CNF sampai bisa diputuskan apakah formula tersebut *satisfiabel* atau *unsatisfiabel.*

Algoritma Berikut ini adalah 5 aturan dasar tersebut :

1. Unit propagation

Unit Propagation merupakan teknik penyederhanaan yang penting dalam suatu proposisiyang berbentuk CNF.Aturan Unit propagation hanya dapat diterapkan jika terdapat klausa yang terdiri dari hanya 1 literal dalam formula. Proposisi pada literal itu ditugasi dengan nilai kebenaran sehingga bernilai *True.* Dengan penugasan ini semua klausa yang memuat literal yang sama akan bernilai *True* sehingga klausa-klausa tersebut bisa dihapus dari formula. Literal yang bernegasi dengan literal tersebut dihapus dari klausa yang memuatnya.Sebagai contoh dapat dilihat pada Gambar 2.1 berikut (Taufiq dan Bahariyanto, 2018).

$$S = \{P \vee R \vee Q\} \wedge \{\sim Q \vee \sim P\} \wedge Q \wedge \sim R$$

Gambar 2.1Contoh soal logika proposisi *unit propagation.*

Terapkan *Unit propogation* dengan L= Q, maka klausa yang memiliki literal Q harus dihapus, seperti yang ditunjukkan pada Gambar 2.2.

$$S` = \{\sim Q \vee \sim P\} \wedge \sim R$$

Gambar 2.2Penerapan *unit propagation* pada logka proposisi (Hapus Klausa).

Hapus ~L = ~Q dari klausa S`, maka setiap literal ~Q harus dihapus, seperti yang ditunjukkan pada Gambar 2.3.

$$S`` = \sim P \wedge \sim R$$

Gambar 2.3Penerapan lanjut *unit propogation* pada logika proposisi (Hapus literal).

2. Pure literal

Aturan Pure literaldapat diterapkan jika terdapat sebuah literal yang hanya muncul dalam satu bentuk saja sementara negasi literal tersebut tidak muncul di klausa manapun. Literal tersebut ditugasi dengan nilai kebenaran sehingga bernilai T dan semua klausa yang memuat literal tersebut dihapus dari formula logika proposisi yang diberikan seperti yang ditunjukkan pada Gambar 2.4.

$$\varphi = (\sim x1 \vee x2) \wedge (x3 \vee \sim x2) \wedge (x4 \vee \sim x5) \wedge (x5 \vee \sim x4)$$

Gambar 2.4Contoh soal logika proposisi pure literal.

Pada proposisi di atas, x1 dan x3 merupakan *Pure Literal*.Maka dari itu hasil dai penerapan fungsi *Pure Literal* pada proposisi diatas ditunjukkan pada gambar 2.5.

$$\varphi = (x4 \vee \sim x5) \wedge (x5 \vee \sim x4)$$

Gambar 2.5Penerapan *pure literal* pada logika proposisi.

3. Decide

Fungsi *decide* juga sering disebut sebgai splitting dan dilakukan apabila dalam suatu proposisi sudah tidak bisa lagi diselesaikan dengan *Unit propagation* dan *Pure literal.* Aturan Decide dapat diterapkan jika semua literal tampil dalam 2 bentuk, misalnya p dan ¬p dan tampil dalam klausa yang berbeda. Jika ini muncul maka yang harus dilakukan adalah menugaskan salah satu literal dengan T, misalnya p, menghapus semua klausa yang memuat p, dan menghapus semua literal ¬p. Jika penugasan ini gagal *(fail)* maka diganti dengan penugasan negasinya (¬p). Contoh Penggunaan fungsi *decide* ditunjukkan pada Gambar 2.6

$$\Rightarrow \quad (p \vee \sim q) \wedge (\sim p \vee q) \wedge (\sim p \vee \sim q) \quad \text{(Decide p=false)}$$
$$\Rightarrow \quad \sim q \quad \text{(Unit Propagation)}$$
$$\Rightarrow \quad \top \quad \text{(satisfiable)}$$

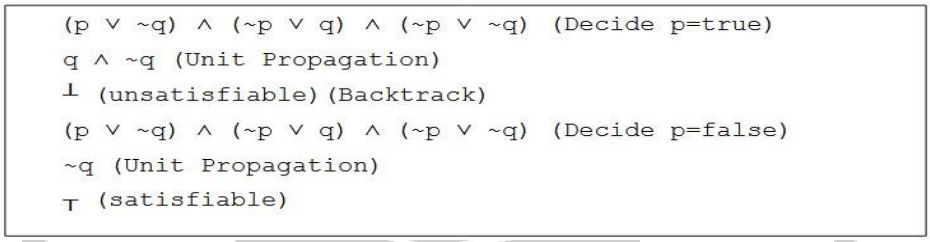
Gambar 2.6Contoh penerapan fungsi *decide*.

4. Fail (unsatisfiable)

Aturan ini diterapkan jika terdapat klausa yang kosong. Jika sebelumnya tidak ada aturan *decide*yang diberlakukan maka formula *unsatisfiable.*

5. Backtrack

Aturan backtrack dapat diterapkan jika terjadi*fail* dan muncul di dalam *decide*. Sehingga yang harus dilakukan adalah kembali ke *decide* terdekat dan mengganti penugasan literal dengan negasinya. Misal, sebelumnya p yang ditugasi, maka langkah selanjutnya adalah menugakan¬p .contoh dari penggunaan fungsi *backtracking* dapat dilihat pada gambar 2.7



Gambar 2.7Contoh penerapan fungsi *backtracking*

d. AHK Script

AHKScript adalah perangkat lunak (*software)* yang awalnya dimaksudkan untuk mengubah hotkey kustom ke tindakan yang berbeda tetapi sekarang merupakan suite otomatisasi Windows lengkap.

Script disini sendiri berarti sebuah kumpulan perintah yang dapat berupa sebuah *Macro*, *Hotkey* ataupun kumpulan perintah seperti program pada umumnya.

AHKScript memiliki konsep yang cukup sederhana, tetapi merupakan bahasa pemrograman Turing-complete yang cukup lengkap.

Code AHK dapat dibuat pendek atau panjang sesuai kebutuhan, tak ada batasan khusus dalam pembuatan baris-baris code, didalam code juga dapat ditambahkan sebuah komentar dengan terlebih dahulu menambahkan simbol ; (titik koma) di ikuti komentar. Sebuah komentar tak akan ikut tereksekusi bersama baris perintah code lain.

Proses instalasi AutoHotkey sangat mudah. Dengan mengunduh penginstal dari situs web resmi danpilih "Instalasi Ekspres." Setelah menginstal perangkat lunak tersebut, kemudian klik kanan di mana saja dan pilih New> AutoHotkey Script untuk membuat skrip baru.