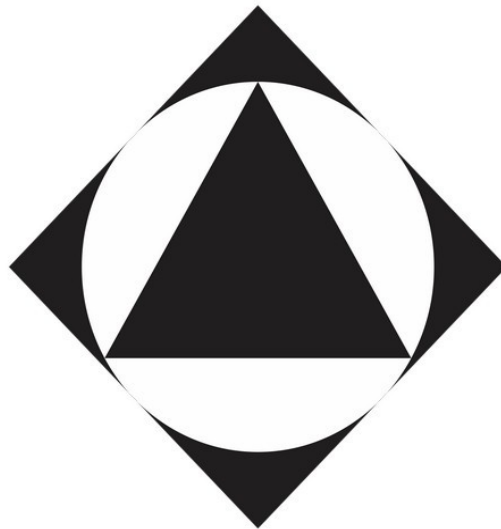


# **LAPORAN**

## **Implementasi Metode Goal Stack Planning dalam Menyelesaikan Block World Problem Berbasis Web**

Disusun untuk memenuhi salah satu tugas mata kuliah IFB-305 Kecerdasan  
Buatan yang diberikan oleh: Youllia Indrawaty Nurhasanah, S.T., M.T.



Disusun oleh:

Kelompok 10

Cikal Gemintang Seya

15-2022-099

Muhammad Ferran Hilmadiansyah

15-2022-102

Kelas AA

**INSTITUT TEKNOLOGI NASIONAL BANDUNG**

**FAKULTAS TEKNOLOGI INDUSTRI**

**INFORMATIKA**

**2022/2023**

## **KATA PENGANTAR**

Puji syukur kehadiran Tuhan Yang Maha Kuasa atas segala limpahan rahmat, taufik, dan hidayah-Nya sehingga kami dapat menyelesaikan penyusunan laporan ini guna memenuhi tugas akhir untuk mata kuliah IFB-305 Kecerdasan Buatan, dengan judul “Implementasi Metode Goal Stack Planning dalam Menyelesaikan Block World Problem Berbasis Web”.

Laporan ini bertujuan untuk menerapkan pemahaman kami dengan salah satu materi pembelajaran mata kuliah IFB-305 Kecerdasan Buatan. Kami menyampaikan ucapan terima kasih kepada Ibu Youllia Indrawaty Nurhasanah, S.T., M.T. selaku dosen mata kuliah yang telah memberikan pelajaran, ilmu, wawasan, serta arahan dalam topik ini. Tidak lupa pula kami untuk berterima kasih kepada semua pihak yang telah membantu dalam penyusunan laporan ini.

Tentu laporan ini tidak lepas dari kesalahan dan kekurangan. Oleh karena itu, kami memohon maaf atas segala kesalahan yang tertulis dalam laporan ini, serta mengharapkan masukan dan saran dari pembaca untuk perbaikan di masa depan.

Bandung, 16 Agustus 2023

Kelompok 10

## DAFTAR ISI

KATA PENGANTAR.....	i
DAFTAR ISI.....	ii
DAFTAR GAMBAR.....	iii
BAB 1 PENDAHULUAN.....	1
1.1. Latar Belakang.....	1
1.2. Rumusan Masalah.....	1
1.3. Batasan Masalah.....	2
1.4. Tujuan.....	2
1.5. Manfaat.....	2
BAB 2 LANDASAN TEORI.....	3
2.1. Pengertian Goal Stack Planning.....	3
2.2. Cara Mengimplementasikan Goal Stack Planning.....	3
2.3. Teknologi yang Digunakan.....	4
2.3.1. HTML.....	4
2.3.2. JavaScript.....	5
2.3.3. Node.js.....	5
2.3.4. TypeScript.....	5
2.4. Pengertian Block World Terhadap Goal Stack Planning.....	6
BAB 3 PERANCANGAN.....	7
3.1. Diagram Blok.....	7
3.2. Flowchart.....	9
3.3. PAGE (Percept, Action, Goal, Environment).....	12
BAB 4 IMPLEMENTASI.....	13
4.1. Kode Sumber.....	13
4.2. Aplikasi.....	17
BAB 5 PENUTUP.....	18
5.1. Kesimpulan.....	18
5.2. Saran.....	18
DAFTAR PUSTAKA.....	19

## DAFTAR GAMBAR

Gambar 1: Tabel PAD (Precondition, Add, Delete).....	6
Gambar 2: Diagram blok proses aplikasi.....	7
Gambar 3: Diagram blok proses Precondition, Add, Delete (PAD).....	8
Gambar 4: Diagram alir proses validasi state.....	9
Gambar 5: Diagram alir fungsi applyPAD.....	10
Gambar 6: Diagram alir fungsi getBlockFromCurrentStates.....	11
Gambar 7: Diagram alir fungsi nextIteration (1).....	11
Gambar 8: Diagram alir fungsi nextIteration (2).....	12
Gambar 9: Diagram alir fungsi nextIteration (3).....	12
Gambar 10: Kode sumber GSP.ts (1).....	13
Gambar 11: Kode sumber GSP.ts (2).....	13
Gambar 12: Kode sumber GSP.ts (3).....	14
Gambar 13: Kode sumber GSP.ts (4).....	14
Gambar 14: Kode sumber index.html.....	15
Gambar 15: Kode sumber script.js (1).....	15
Gambar 16: Kode sumber script.js (2).....	16
Gambar 17: Kode sumber script.js (3).....	16
Gambar 18: Tangkapan layar implementasi aplikasi.....	17
Gambar 19: Tangkapan layar penggunaan aplikasi.....	17

# **BAB 1**

## **PENDAHULUAN**

### **1.1. Latar Belakang**

Block World Problem merupakan salah satu masalah klasik dalam bidang kecerdasan buatan. Masalah ini merupakan masalah perencanaan untuk memanipulasi blok-blok agar mencapai keadaan akhir yang diinginkan. Salah satu metode yang dapat digunakan untuk menyelesaikan Block World Problem adalah metode Goal Stack Planning.

Metode Goal Stack Planning awalnya dikembangkan dalam konteks kecerdasan buatan untuk merencanakan tindakan yang terdiri dari serangkaian langkah yang lebih kecil. Pendekatan ini didasarkan pada ide hierarki tujuan, di mana tujuan yang lebih tinggi dapat dipecah menjadi tujuan-tujuan yang lebih spesifik dan tindakan-tindakan yang konkret. Konsep ini berakar dalam ilmu kognitif manusia, di mana pemikiran dan perencanaan manusia sering kali melibatkan hierarki tujuan.

Oleh karena itu, latar belakang ini merangkum sebuah dorongan untuk mengimplementasikan metode Goal Stack Planning dalam lingkup Block World Problem dengan pendekatan berbasis web, dengan harapan dapat menghasilkan kontribusi signifikan dalam bidang pengembangan teknologi dan penelitian perencanaan yang lebih lanjut terutama dalam bidang pengembangan web.

### **1.2. Rumusan Masalah**

Dari latar belakang di atas maka dapat ditarik beberapa rumusan masalah, yaitu:

1. Bagaimana cara menyelesaikan suatu keadaan tumpukan balok dengan keadaan yang diinginkan?
2. Bagaimana cara mengimplementasikan Goal Stack Planning untuk menyelesaikan Block World Problem?
3. Bagaimana cara memvisualisasikan cara penyelesaian Block World Problem dengan metode Goal Stack Planning dalam suatu aplikasi web?

### **1.3. Batasan Masalah**

Dikarenakan lingkup perencanaan kecerdasan buatan dan pengembangan web yang sangat luas, batasan masalah yang dibuatkan untuk laporan ini ialah:

1. Metode Goal Stack Planning adalah sebagaimana yang telah dipelajari dalam mata kuliah IFB-305 Kecerdasan Buatan.
2. Perancangan antarmuka web digunakan untuk memfasilitasi interaksi pengguna, namun tidak melibatkan visualisasi yang rumit.
3. Alternatif penyelesaian dan optimisasi algoritma tidak akan menjadi fokus utama dalam laporan ini.

### **1.4. Tujuan**

Ada pula tujuan dari laporan ini, yaitu:

1. Implementasi metode Goal Stack Planning untuk menyelesaikan Block World Problem.
2. Integrasi antarmuka dengan algoritma perencanaan metode Goal Stack Planning dalam lingkungan web.

### **1.5. Manfaat**

Dari laporan ini bisa didapatkan manfaat untuk umum, khususnya bagi mahasiswa yang akan mempelajari mata kuliah IFB-305 Kecerdasan Buatan, yakni:

1. Pembelajaran interaktif untuk memahami lebih lanjut tentang cara kerja metode Goal Stack Planning dalam menyelesaikan Block World Problem.
2. Penghematan waktu dan upaya dalam penyelesaian Block World Problem dengan menggunakan algoritma pemecah masalah otomatis.

## **BAB 2**

### **LANDASAN TEORI**

#### **2.1. Pengertian Goal Stack Planning**

Goal Stack Planning (GSP) adalah sebuah pendekatan dalam bidang kecerdasan buatan (*artificial intelligence*) yang digunakan untuk perencanaan tindakan. Tujuan utama dari GSP adalah merencanakan urutan tindakan yang diperlukan untuk mencapai tujuan tertentu. Pendekatan ini fokus pada representasi hierarkis dari tujuan dan tindakan yang diperlukan untuk mencapai tujuan tersebut.

Dalam GSP, tujuan dan tindakan diwakili dalam bentuk tumpukan (*stack*). Pada awalnya, tumpukan hanya berisi tujuan akhir yang ingin dicapai. Kemudian, tindakan-tindakan yang diperlukan untuk mencapai tujuan tersebut secara bertahap ditambahkan ke tumpukan. Setiap tindakan bisa memiliki sub-tindakan atau subtujuan sendiri, dan tumpukan secara perlahan terisi dengan hierarki tindakan yang semakin rinci.

GSP membantu dalam merencanakan urutan tindakan dengan cara yang sistematis dan terstruktur. Pendekatan ini memungkinkan sistem untuk menguraikan tujuan kompleks menjadi serangkaian tindakan yang lebih sederhana, memperhitungkan ketergantungan antara tindakan-tindakan tersebut. Selain itu, GSP juga dapat menangani situasi di mana beberapa tujuan harus dicapai secara bersamaan atau berurutan.

Namun, perlu dicatat bahwa GSP adalah salah satu dari banyak pendekatan yang ada dalam bidang perencanaan kecerdasan buatan, dan ada pendekatan lain seperti Planning Graphs, STRIPS, dan lain-lain yang digunakan untuk mengatasi masalah perencanaan.

#### **2.2. Cara Mengimplementasikan Goal Stack Planning**

Mengimplementasikan Goal Stack Planning (GSP) melibatkan pembuatan program komputer yang dapat merencanakan tindakan (*operation*) untuk mencapai tujuan tertentu menggunakan pendekatan tumpukan tujuan (*goal stack*) dan *operation*.

Di bawah ini adalah hal yang perlu disiapkan untuk mengimplementasikan GSP:

1. Struktur data yang akan digunakan untuk merepresentasikan kondisi (*state*), tumpukan (*stack*), dan baris (*queue*).

2. Mendefinisikan kondisi awal (*initial state*) dan kondisi tujuan (*goal state*) menggunakan struktur data yang telah dibuat secara hierarkis sehingga tidak terjadi konflik di antara kedua kondisi.
3. Membuat fungsi dan prosedur yang akan dibutuhkan untuk memanipulasi kondisi dan tumpukan (*stack*) sehingga tidak memenuhi kode sumber dengan baris yang terulang-ulang.

Setelah itu, mulailah dengan menerapkan logika GSP. Berikut adalah langkah umum yang dapat diikuti dalam implementasi:

- a) Inisialisasi tumpukan dengan tujuan akhir.
- b) Selama tumpukan tujuan tidak kosong:
  - Ambil tujuan teratas dari tumpukan.
  - Jika tujuan tersebut sudah tercapai atau merupakan tindakan, lanjutkan ke tujuan berikutnya.
  - Jika tujuan tersebut adalah tujuan yang kompleks, tambahkan subtujuan atau tindakan yang diperlukan ke tumpukan.

Mengimplementasikan GSP memerlukan pemahaman yang kuat tentang konsep perencanaan kecerdasan buatan dan kemampuan pemrograman. Tidak disarankan untuk yang baru mengenal konsep ini karena memerlukan waktu untuk memahami prinsip-prinsip dasar sebelum memulai implementasi.

## **2.3. Teknologi yang Digunakan**

### **2.3.1. HTML**

Menurut Hidayatullah dan Kawistara (2015) dalam jurnal Fitri Ayu dan Nia Permata Sari (ISSN:2549-0222) “Hypertext Markup Language (HTML) adalah bahasa standar yang digunakan untuk menampilkan halaman web”.

HTML5 (Hypertext Markup Language version 5) adalah sebuah bahasa markah yang menstrukturkan isi dari World Wide Web, sebuah teknologi utama pada internet. Standar HTML5 menyempurnakan elemen-elemen lama yang terdapat pada standar sebelumnya, menambahkan elemen-elemen yang lebih semantik dan menambahkan fitur-fitur baru untuk mendukung pembuatan aplikasi web yang lebih kompleks.



### 2.3.2. JavaScript

JavaScript adalah bahasa pemrograman tingkat tinggi yang paling umum digunakan dalam pengembangan web. JavaScript digunakan untuk mengontrol perilaku interaktif pada sisi klien (browser) dalam pengembangan web. JavaScript umumnya dieksekusi pada browser klien dan digunakan untuk membuat tampilan dinamis, mengelola interaksi pengguna, dan melakukan permintaan ke server.

### 2.3.3. Node.js

Node.js adalah sistem perangkat lunak dan lingkungan *runtime* yang dibangun di atas mesin JavaScript V8 milik Google sehingga memungkinkan eksekusi JavaScript di sisi server. Node.js memungkinkan pengembang untuk membuat aplikasi baik *server-side* ataupun *client-side* menggunakan JavaScript, mengatasi banyak permintaan secara efisien, dan mengelola operasi asynchronous melalui event loop.

### 2.3.4. TypeScript

TypeScript adalah sebuah bahasa pemrograman sumber terbuka (*open source*) yang dikembangkan oleh Microsoft. TypeScript adalah superset dari Javascript, artinya TypeScript memiliki semua fitur yang ada pada JavaScript dengan tambahan beberapa fitur. Beberapa fitur tambahan pada TypeScript adalah dukungan pemrograman berorientasi objek dan tipe statis secara opsional. TypeScript ini dikembangkan dengan tujuan untuk membuat aplikasi dalam skala besar dan akan dikompilasi ke bahasa JavaScript.

TypeScript dan Node.js saling terkait karena TypeScript menghasilkan kode JavaScript. Dalam pengembangan aplikasi Node.js menggunakan TypeScript, kode yang ditulis dalam bahasa TypeScript kemudian dikompilasi menjadi JavaScript agar bisa dieksekusi oleh Node.js. Penggunaan TypeScript dalam pengembangan Node.js membantu dalam mengurangi kesalahan dan meningkatkan kualitas kode.

## 2.4. Pengertian Block World Terhadap Goal Stack Planning

Dalam block world memiliki beberapa balok yang ditempatkan di atas meja. Tujuan Anda adalah mengatur balok-blok ini dalam urutan tertentu. Konsep Goal Stack Planning digunakan untuk merencanakan langkah-langkah yang diperlukan untuk mencapai tujuan ini. Di dalam dunia balok terdapat nama istilah yaitu:

1. Initial State: Keadaan awal balok.
2. Goal State: Tujuan balok yang diinginkan
3. Operator: Langkah-langkah konkret yang dapat diambil untuk mencapai tujuan dalam lingkungan. Misalnya, dalam "dunia balok," beberapa operator mungkin melibatkan:
  - a) PICKUP: Mengambil balok dari posisi di atas meja.
  - b) PUTDOWN: Meletakkan balok pada posisi yang diinginkan di atas meja.
  - c) STACK: Menumpuk balok satu di atas balok lainnya.
  - d) UNSTACK: Mengambil balok dari posisi balok di atas balok.
4. Arm: Lengan robot untuk memanipulasi balok.

Di dalam goal stack planning terdapat nama istilah juga, yaitu:

1. Stack: Tumpukan untuk menampung States.
2. Current State: Kondisi saat ini.
3. Queue: Antrian untuk menampung solusi.

Singkatan PAD ini berasal dari P (Pre-condition), A (Add), dan D (Delete).

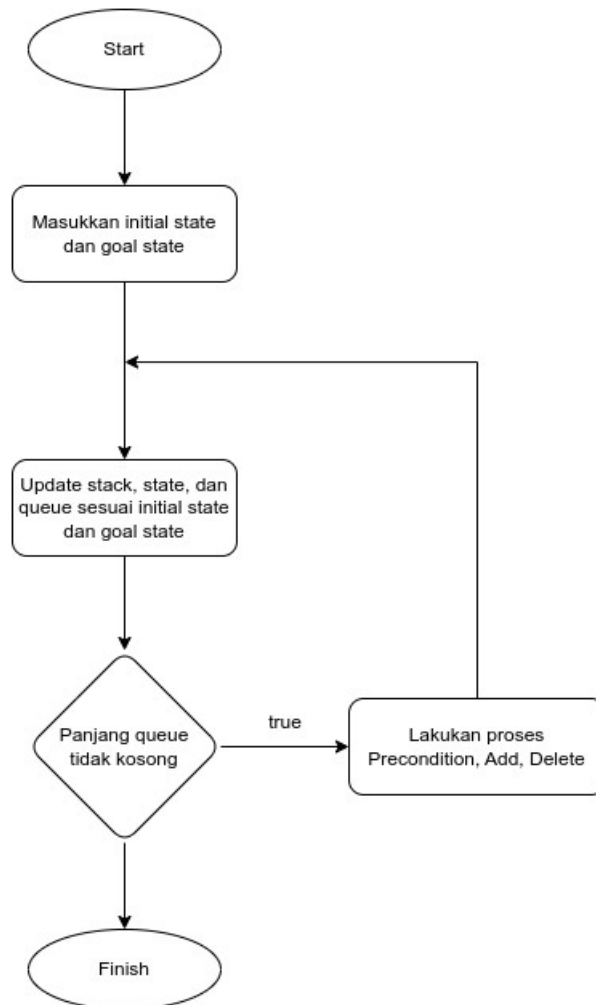
STACK(A,B)	
P	: HOLDING(A) $\wedge$ CLEAR(B)
A	: ON(A,B) $\wedge$ ARMEMPTY
D	: HOLDING(A) $\wedge$ CLEAR(B)
UNSTACK(A,B)	
P	: ON(A,B) $\wedge$ CLEAR(A) $\wedge$ ARMEMPTY
A	: HOLDING(A) $\wedge$ CLEAR(B)
D	: ON(A,B) $\wedge$ ARMEMPTY
PICKUP(A)	
P	: ONTABLE(A) $\wedge$ CLEAR(A) $\wedge$ ARMEMPTY
A	: HOLDING(A)
D	: ONTABLE(A) $\wedge$ ARMEMPTY
PUTDOWN(A)	
P	: HOLDING(A)
A	: ONTABLE(A) $\wedge$ ARMEMPTY
D	: HOLDING(A)

Gambar 1: Tabel PAD (Precondition, Add, Delete)

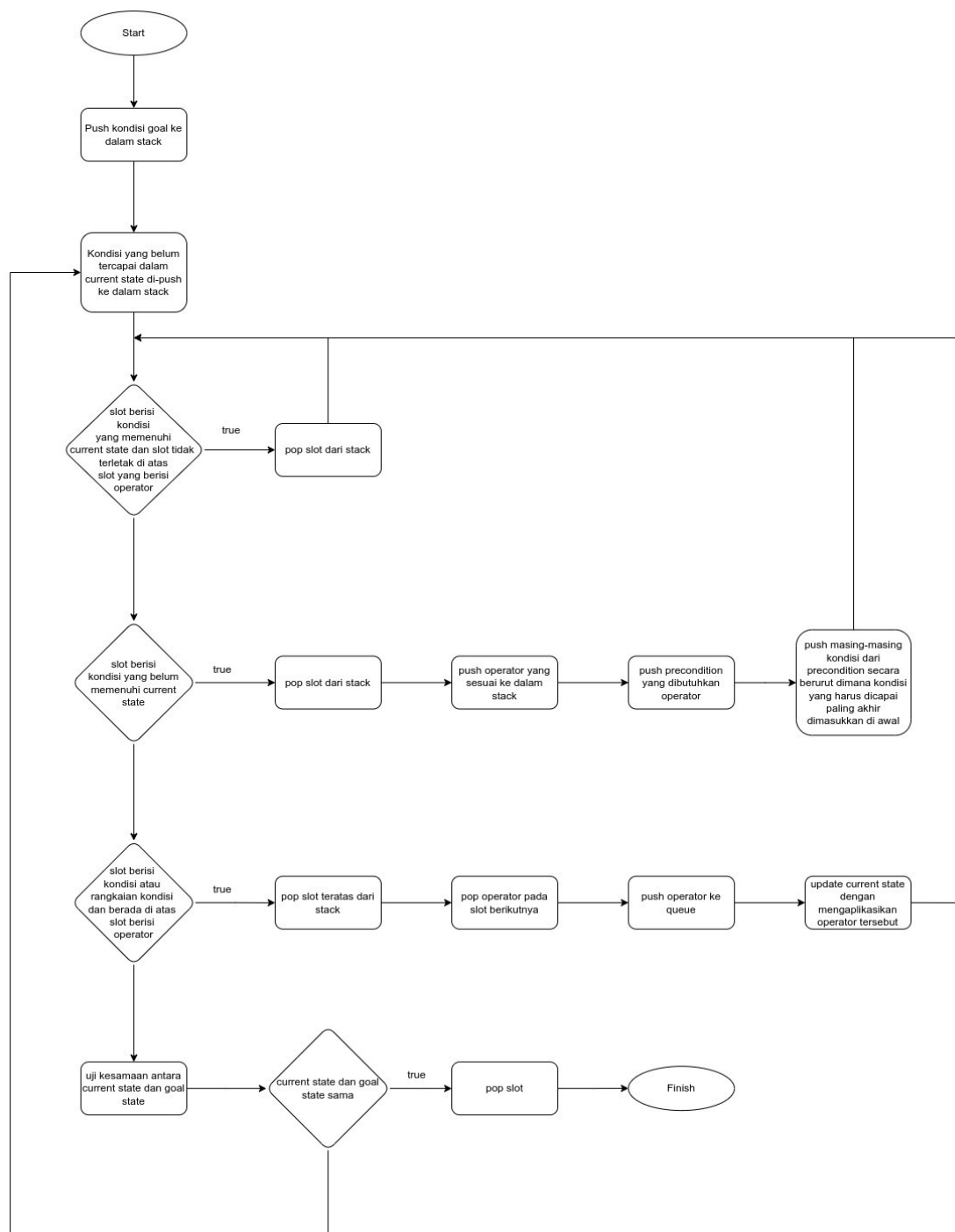
## BAB 3

### PERANCANGAN

#### 3.1. Diagram Blok

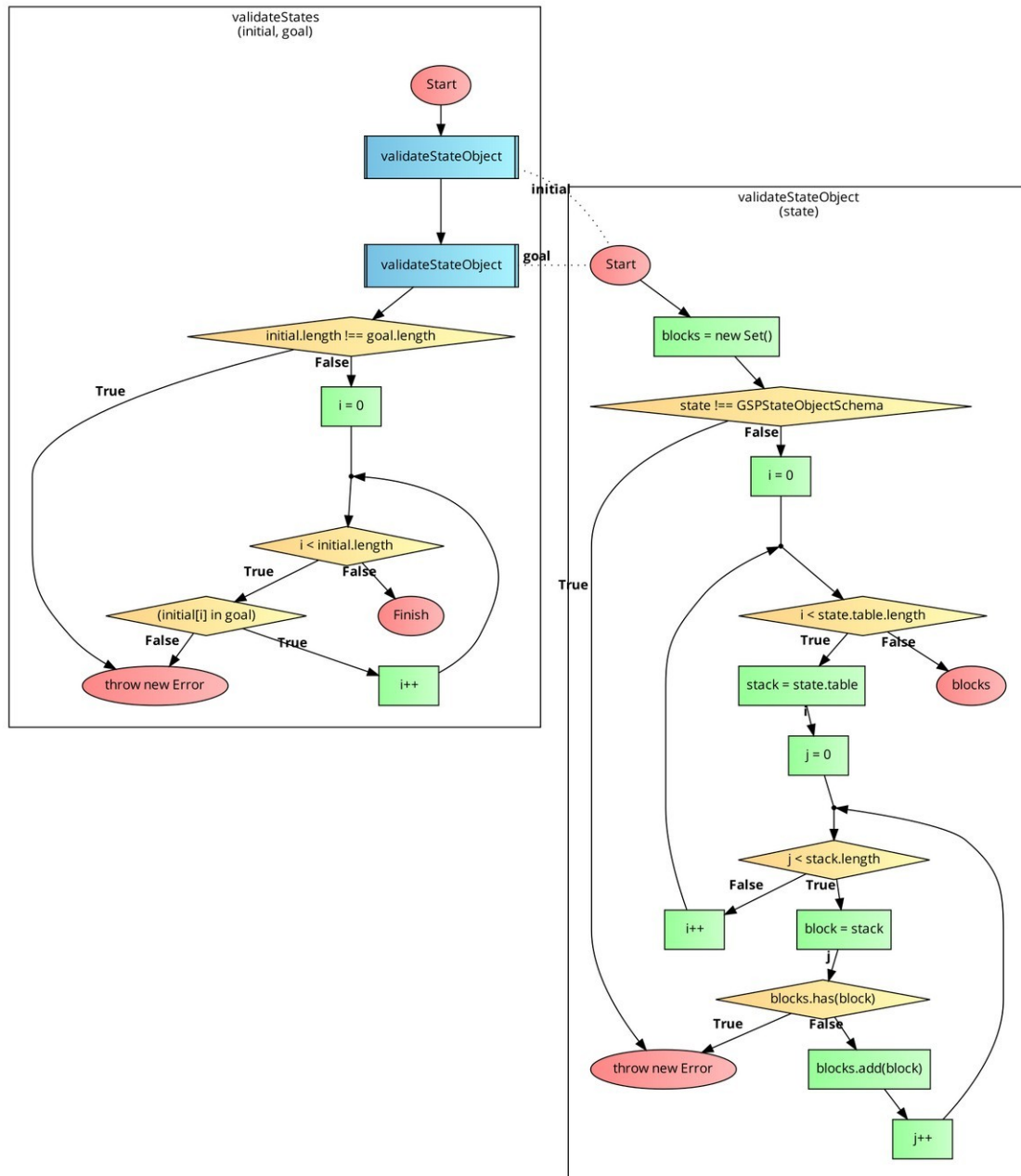


Gambar 2: Diagram blok proses aplikasi

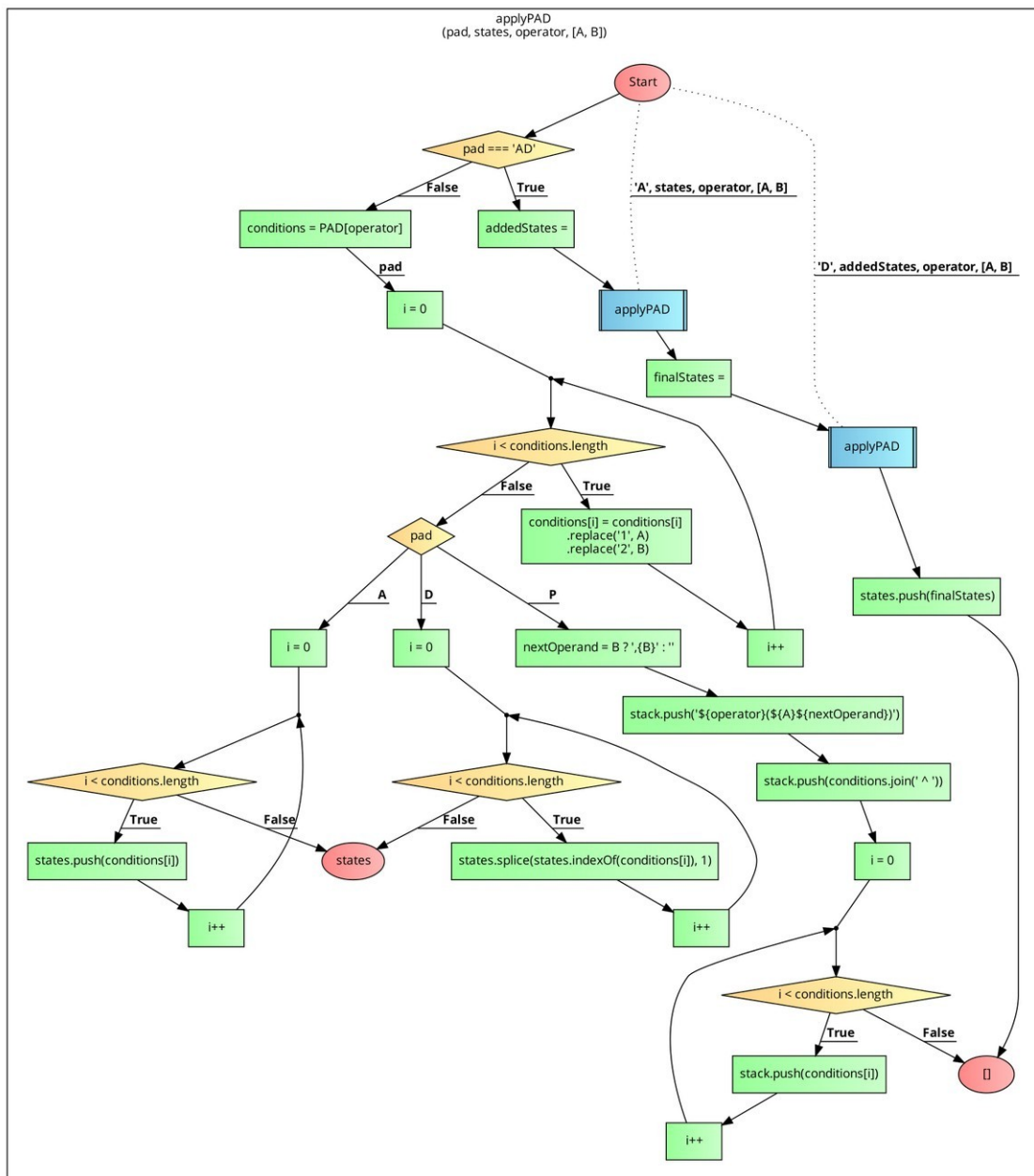


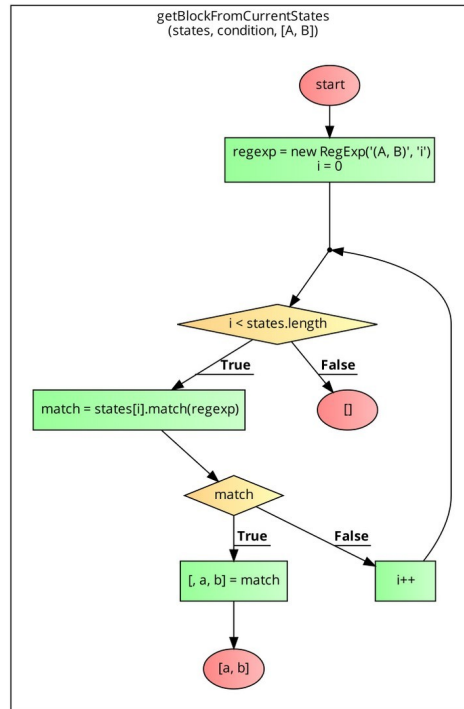
Gambar 3: Diagram blok proses Precondition, Add, Delete (PAD)

### 3.2. Flowchart

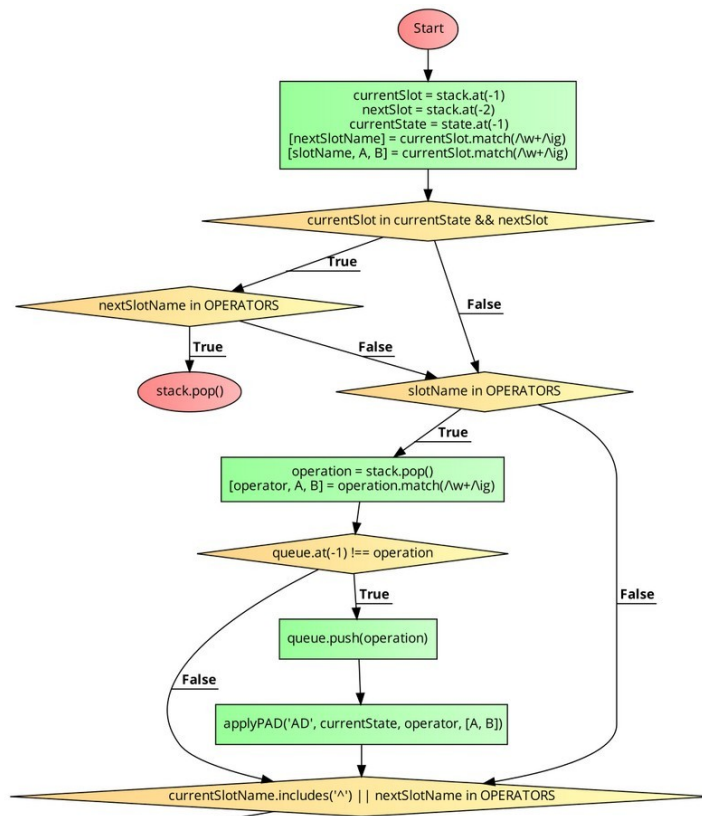


Gambar 4: Diagram alir proses validasi state

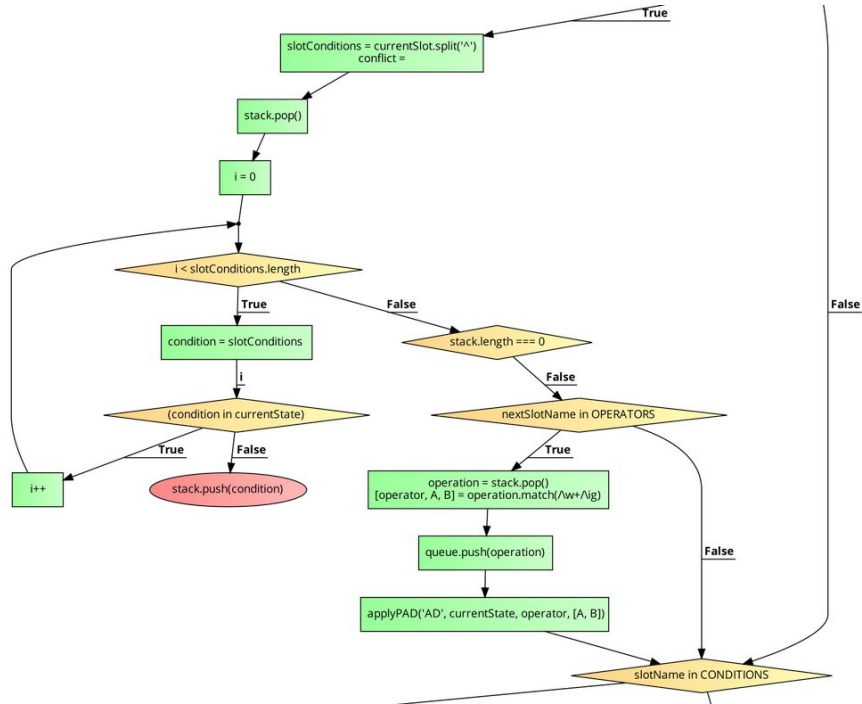




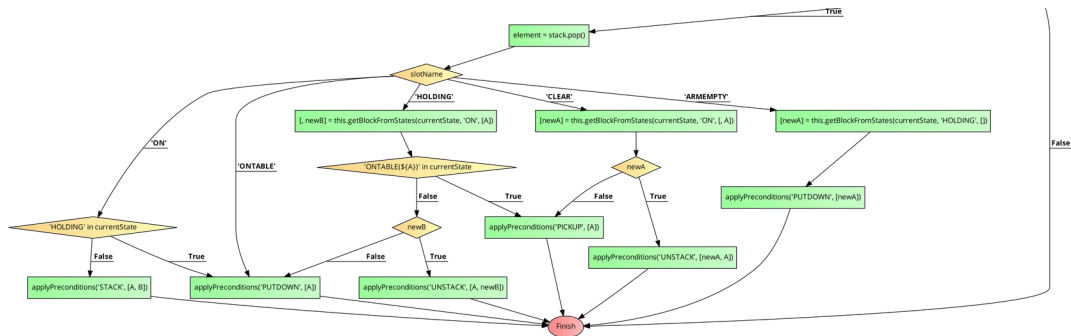
Gambar 6: Diagram alir fungsi  
getBlockFromCurrentStates



Gambar 7: Diagram alir fungsi nextIteration (1)



Gambar 8: Diagram alir fungsi nextIteration (2)



Gambar 9: Diagram alir fungsi nextIteration (3)

### 3.3. PAGE (Percept, Action, Goal, Environment)

- Agent Type: Block Puzzle Solver
- Percept: Clear, On, Ontable, Holding, Armempty
- Action: Stack, Unstack, Pickup, Putdown
- Goal: Menyesuaikan initial state dengan goal state
- Environment: Dasar meja, block, arm



## IMPLEMENTASI

#### 4.1. Kode Sumber

Semua kode sumber tercantum dan tak tercantum di laporan ini dapat diakses dalam repositori GitHub dengan link <https://github.com/sglkc/goal-stack-planning>.

```

1 export type GSPStateObject = {
2   tables: string[];
3   arms: string | null | undefined
4 }
5
6 type GSPCondition = typeof GSP_CONDITIONS[number]
7 type GSPOperator = typeof GSP_OPERATIONS[number]
8 type GSPADPType = Record<
9   GSPOperator,
10   Record<P, 'A' | 'D', string[]>
11 >
12 export type GSPStateOptions = {
13   draw?: boolean
14   logging?: boolean
15 }
16
17 export type GSPIterationReturn = void | {
18   ops: 'push' | 'pop'
19   object: 'stack' | 'queue' | 'state'
20   element: string
21 }
22
23 export default class GSP {
24   // Konstanta kondisi pada stack
25   static CONDITIONS = ['CLEAR', 'ON', 'ONTABLE', 'HOLDING', 'ARMEMPTY'] as const
26   static OPERATIONS = ['STACK', 'UNSTACK', 'PICKUP', 'PUTDOWN'] as const
27   static PAD: GSPADPType = {
28     STACK: {
29       P: ['CLEAR(2)', 'HOLDING(1)'],
30       A: ['ON(1,2)', 'CLEAR(1)', 'ARMEMPTY'],
31       D: ['CLEAR(2)', 'HOLDING(1)'],
32     },
33     UNSTACK: {
34       P: ['ON(1,2)', 'CLEAR(1)', 'ARMEMPTY'],
35       A: ['CLEAR(2)', 'HOLDING(1)'],
36       D: ['ON(1,2)', 'CLEAR(1)', 'ARMEMPTY'],
37     },
38     PICKUP: {
39       P: ['ONTABLE(1)', 'CLEAR(1)', 'ARMEMPTY'],
40       A: ['HOLDING(1)'],
41       D: ['ONTABLE(1)', 'CLEAR(1)', 'ARMEMPTY'],
42     },
43     PUTDOWN: {
44       P: ['HOLDING(1)'],
45       A: ['HOLDING(1)', 'CLEAR(1)', 'ARMEMPTY'],
46       D: ['ONTABLE(1)', 'CLEAR(1)', 'ARMEMPTY'],
47     },
48   },
49   // Progress menyelesaikan
50   state: string[] = []
51   stack: string[] = []
52   queue: string[] = []
53 }
54
55 // Atribut yang bisa diubah user
56 GSP.ts
57
58 // Atribut yang bisa diubah user
59 1 maxIterations: number = 30
60 2 initial: GSPStateObject
61 3 goal: GSPStateObject
62
63 // Di parameter untuk meminimalisasi new GSP(initial, goal, 10)
64 constructor(initial: GSPStateObject, goal: GSPStateObject,
65   maxIterations: number = 30) {
66   this.maxIterations = maxIterations
67   this.initial = Object.assign({}, initial)
68   this.goal = Object.assign({}, goal)
69   this.validateStates()
70 }
71
72 // Method static private utk validasi objek state sebelum
73 di proses
74 static validateStateObject(state: GSPStateObject, name =
75   'stack'): Set<string> {
76   const blocks = new Set<string>()
77
78   if (typeof state !== 'object') throw new Error('$($name) harus
79     berupa object')
80
81   if (Array.isArray(state.tables)) throw new Error('$($name)
82     table harus berupa array')
83
84   if (typeof state.arms === 'string' | null, undefined)
85     includeStateObject(state, 'stack', 'stack', 'this.goal')
86   else throw new Error('$($name).arms harus berupa string, null,
87     atau undefined')
88
89   state.table.forEach((stack) => {
90     stack.forEach((block) => {
91       if (blocks.has(stack.arms)) throw new Error('$($name) block
92         $($block) ada duplikat')
93       blocks.add(block)
94     })
95   })
96
97   if (typeof state.arms === 'string') {
98     if (blocks.has(state.arms)) throw new Error('$($name) block
99       $($state.arms) ada di table dan arms')
100     blocks.add(state.arms)
101   }
102
103   return blocks
104 }
105
106 // Print objek state ke console dengan format
107 static printStateObject(state: GSPStateObject, name?: string):
108   void {
109   if (!this.validateStateObject(state))
110     const cons = {}
111     let height = 0
112     if (name) console.log `%c` + name, 'color: white; font-
113       weight: bold; font-size: 1.2em;`
114
115   scene.push('ARM: ' + (state.arms ? 'KOSONG' : ''))
116
117   // Cari tumpukan dengan block terbanyak (tertinggi)
118   state.table.forEach((t) => {
119     height = (length > height) ? t.length : height
120   })
121
122   // Gambar block di setiap tumpukan di tingkatan yang sama
123   for (height; height >= 0; height--) {
124     const l = height + 1
125     const level = state.table.map(t => t[l ? 1 : 0] ? t[l] : undefined)
126     t.iter(t => t)
127     scene.push(level.join(' '))
128   }
129
130 // But Tunggu dan print ke console
131 scene.push('...repeat @ state.table.length - 1)')
132 console.log `%c` + scene.join('\n'), 'font-size: 0.8em')
133 }
134
135 // Cek jika initial state dan goal state tidak valid
136 validateStates() {
137   const initialBlocks = GSP.validateStateObject(this.initial, 'this.initial')
138   const goalBlocks = GSP.validateStateObject(this.goal, 'this.goal')
139
140   if (initialBlocks.size !== goalBlocks.size)
141     throw new Error('Initial state dan goal state memiliki block yang berbeda')
142
143   initialBlocks.forEach((block) => {
144     if (!goalBlocks.has(block))
145       throw new Error('Initial state dan goal state memiliki block yang berbeda')
146   })
147
148 // Generate array kondisi dari objek state
149 static generateStateConditions(state: GSPStateObject): string[] {
150   const conditions: string[] = []
151
152   state.table.forEach((stack) => {
153     // Reverse array karena data struktur stack dimulai dari akhir
154     const reversedStack = Array.from(stack, reverse)
155
156     reversedStack.forEach((block, blockIndex) => {
157       if (blockIndex === reversedStack.length - 1)
158         conditions.push('ONTABLE($block)')
159       else
160         conditions.push('CLEAR($block)')
161       return
162     })
163
164     const nextBlock = reversedStack.at(blockIndex + 1)
165     const conditionPush = 'ON($block), $nextBlock()')
166   })
167 }
168
169 NORMAL src GSP.ts * master /n typescript UTF-8 101.3 C

```

Gambar 10: Kode sumber GSP.ts (1)

```

1577         if (blockIndex === 0) conditions.push('CLEAR${block}');
1578     }
1579 }
1580
1581 if (state.ari) conditions.push('HOLDING${state.ari}');
1582 else conditions.push('ARMEMPTY');
1583
1584 return conditions
1585
1586 // Membuat state object dari state array string
1587 static generateStateObject(states: string[]): GSPStateObject {
1588     const statesSet: Set<string> = new Set()
1589     const sorted = states.sort((a, b) => a.localeCompare(b))
1590     const object: GSPStateObject = {
1591         areas: null,
1592         table: []
1593     }
1594
1595     sorted.forEach(state => {
1596         const [operator, A, B] = state.match(/\w+(?![\w/]) as RegExpMatchArray
1597
1598         switch (operator) {
1599             case 'ARMEMPTY': break
1600             case 'HOLDING':
1601                 object.aria = A
1602                 break
1603             case 'TABLE':
1604                 object.table.push(A)
1605                 break
1606             case 'ON':
1607                 stacks.add([A, B])
1608                 break
1609         }
1610     })
1611
1612     while (stacks.size())
1613         stacks.forEach(stack => {
1614             const [A, B] = stack
1615             const found = object.table.find(tb) => tb.at(1) === B
1616
1617             if (found) {
1618                 stacks.push(A)
1619                 stacks.delete(stack)
1620             }
1621         })
1622
1623     return object
1624 }
1625
1626 // Membuat state object dari state saat ini
1627 getCurrentStateObject(): GSPStateObject {
1628     const currentState = this.state.at(1)
1629     return currentState ? GSP.generateStateObject(currentState) : this
1630 }
1631
1632 init() {
1633     GSP.ts
1634
1635     getBlockFromStates()
1636     states: string[]
1637     conditions: GSPCondition,
1638     [A, B]: Array<string> | undefined =
1639     Array<string> | undefined =
1640     const matcher = A
1641     ? $(condition)\${A}'
1642     :
1643     ? $(condition)\.\\.\${B}\.\\
1644     : $(condition)
1645
1646     const regexp = new RegExp(matcher, 'i')
1647     const stateMatch = states.find(state => state.match(regexp))
1648
1649     if (stateMatch) return {
1650         [a, b] = stateMatch.match(/\w+(?![\w/]) as RegExpMatchArray
1651     }
1652     return [a, b]
1653 }
1654
1655 applyPAD() {
1656     pad['P' | 'A' | 'D' | 'AD',
1657         states: typeOf this.state, stateNumber,
1658         operator: GSPOperator,
1659         [A, B]: string[]
1660     ] = string[]
1661
1662     const newStates = Array.from(states)
1663
1664     if (pad === 'AD') {
1665         const addedStates = this.applyPAD('A', states, operator, [A, B])
1666         const finalStates = this.applyPAD('D', addedStates, operator, [A, B])
1667
1668         this.state.push(finalStates)
1669         return []
1670     }
1671
1672     const conditions = GSP.PAD.operator[`${pad}`].map((condition) => {
1673         return condition.replace('1', A).replace('2', B)
1674     })
1675
1676     switch (pad) {
1677         case 'P':
1678             const nextOperand = B ? '2', `${B}` : ''
1679
1680             this.state.push(`${operator}(${states}${nextOperand})')
1681             this.state.push(conditions.join('& '))
1682             conditions.forEach(condition => this.state.push(condition))
1683         case 'A':
1684             return []
1685         case 'D':
1686             conditions.forEach(condition => newStates.push(condition))
1687             return newStates
1688         case 'D':
1689             conditions.forEach(condition => {
1690
1691 case 'D'
1692             conditions.forEach(condition) => {
1693                 newStates.splice(newStates.indexOf(condition), 1)
1694                 return newStates
1695             }
1696
1697 // Print STATE, STACK, dan QUEUE saat ini
1698 printCurrentIteration(): void {
1699     const colorizer = 'color: white; background-color: royalblue; font-size: 1em'
1700     const stack = Array.from(this.state)
1701     console.log(
1702         '%STACK, colorizer,
1703         '\n' + (this_stack.length ? stack : '> STACK KOSONG')
1704     )
1705
1706     const state = this.state
1707     .map((item) => item.sort((a, b) => b.localeCompare(a)))
1708     .map((item, i) => `${(i + 1)} + item.join(' ')`)
1709     .join('\n')
1710     .trim()
1711
1712     console.log(
1713         '%STATE, colorizer,
1714         '\n' + (this_state.length ? state : '> STATE KOSONG')
1715     )
1716
1717     const queue = this.queue
1718     .map((item, i) => `${(i + 1)} + ${item}`)
1719     .join('\n')
1720     .trim()
1721
1722     console.log(
1723         '%QUEUE, colorizer,
1724         '\n' + (this_queue.length ? queue : '> QUEUE KOSONG')
1725     )
1726 }
1727
1728 // Reset state, stack, queue dan masukkan kan dan stack dari user
1729 prepare() {
1730     void
1731     this.validateStates()
1732
1733     const initialConditions = GSP.generateStateConditions(this.initial)
1734     const goalConditions = GSP.generateStateConditions(this.goal)
1735
1736     this.state = []
1737     this_stack = []
1738     this_queue = []
1739     this.validateStates()
1740     this.validateStates()
1741     this.validateStates()
1742     this.validateStates()
1743     this.validateStates()
1744     this.validateStates()
1745     this.validateStates()
1746     this.validateStates()
1747     this.validateStates()
1748     this.validateStates()
1749     this.validateStates()
1750     this.validateStates()
1751     this.validateStates()
1752     this.validateStates()
1753     this.validateStates()
1754     this.validateStates()
1755     this.validateStates()
1756     this.validateStates()
1757     this.validateStates()
1758     this.validateStates()
1759     this.validateStates()
1760     this.validateStates()
1761     this.validateStates()
1762     this.validateStates()
1763     this.validateStates()
1764     this.validateStates()
1765     this.validateStates()
1766     this.validateStates()
1767     this.validateStates()
1768     this.validateStates()
1769     this.validateStates()
1770     this.validateStates()
1771     this.validateStates()
1772     this.validateStates()
1773     this.validateStates()
1774     this.validateStates()
1775     this.validateStates()
1776     this.validateStates()
1777     this.validateStates()
1778     this.validateStates()
1779     this.validateStates()
1780     this.validateStates()
1781     this.validateStates()
1782     this.validateStates()
1783     this.validateStates()
1784     this.validateStates()
1785     this.validateStates()
1786     this.validateStates()
1787     this.validateStates()
1788     this.validateStates()
1789     this.validateStates()
1790     this.validateStates()
1791     this.validateStates()
1792     this.validateStates()
1793     this.validateStates()
1794     this.validateStates()
1795     this.validateStates()
1796     this.validateStates()
1797     this.validateStates()
1798     this.validateStates()
1799     this.validateStates()
1800     this.validateStates()
1801     this.validateStates()
1802     this.validateStates()
1803     this.validateStates()
1804     this.validateStates()
1805     this.validateStates()
1806     this.validateStates()
1807     this.validateStates()
1808     this.validateStates()
1809     this.validateStates()
1810     this.validateStates()
1811     this.validateStates()
1812     this.validateStates()
1813     this.validateStates()
1814     this.validateStates()
1815     this.validateStates()
1816     this.validateStates()
1817     this.validateStates()
1818     this.validateStates()
1819     this.validateStates()
1820     this.validateStates()
1821     this.validateStates()
1822     this.validateStates()
1823     this.validateStates()
1824     this.validateStates()
1825     this.validateStates()
1826     this.validateStates()
1827     this.validateStates()
1828     this.validateStates()
1829     this.validateStates()
1830     this.validateStates()
1831     this.validateStates()
1832     this.validateStates()
1833     this.validateStates()
1834     this.validateStates()
1835     this.validateStates()
1836     this.validateStates()
1837     this.validateStates()
1838     this.validateStates()
1839     this.validateStates()
1840     this.validateStates()
1841     this.validateStates()
1842     this.validateStates()
1843     this.validateStates()
1844     this.validateStates()
1845     this.validateStates()
1846     this.validateStates()
1847     this.validateStates()
1848     this.validateStates()
1849     this.validateStates()
1850     this.validateStates()
1851     this.validateStates()
1852     this.validateStates()
1853     this.validateStates()
1854     this.validateStates()
1855     this.validateStates()
1856     this.validateStates()
1857     this.validateStates()
1858     this.validateStates()
1859     this.validateStates()
1860     this.validateStates()
1861     this.validateStates()
1862     this.validateStates()
1863     this.validateStates()
1864     this.validateStates()
1865     this.validateStates()
1866     this.validateStates()
1867     this.validateStates()
1868     this.validateStates()
1869     this.validateStates()
1870     this.validateStates()
1871     this.validateStates()
1872     this.validateStates()
1873     this.validateStates()
1874     this.validateStates()
1875     this.validateStates()
1876     this.validateStates()
1877     this.validateStates()
1878     this.validateStates()
1879     this.validateStates()
1880     this.validateStates()
1881     this.validateStates()
1882     this.validateStates()
1883     this.validateStates()
1884     this.validateStates()
1885     this.validateStates()
1886     this.validateStates()
1887     this.validateStates()
1888     this.validateStates()
1889     this.validateStates()
1890     this.validateStates()
1891     this.validateStates()
1892     this.validateStates()
1893     this.validateStates()
1894     this.validateStates()
1895     this.validateStates()
1896     this.validateStates()
1897     this.validateStates()
1898     this.validateStates()
1899     this.validateStates()
1900     this.validateStates()
1901     this.validateStates()
1902     this.validateStates()
1903     this.validateStates()
1904     this.validateStates()
1905     this.validateStates()
1906     this.validateStates()
1907     this.validateStates()
1908     this.validateStates()
1909     this.validateStates()
1910     this.validateStates()
1911     this.validateStates()
1912     this.validateStates()
1913     this.validateStates()
1914     this.validateStates()
1915     this.validateStates()
1916     this.validateStates()
1917     this.validateStates()
1918     this.validateStates()
1919     this.validateStates()
1920     this.validateStates()
1921     this.validateStates()
1922     this.validateStates()
1923     this.validateStates()
1924     this.validateStates()
1925     this.validateStates()
1926     this.validateStates()
1927     this.validateStates()
1928     this.validateStates()
1929     this.validateStates()
1930     this.validateStates()
1931     this.validateStates()
1932     this.validateStates()
1933     this.validateStates()
1934     this.validateStates()
1935     this.validateStates()
1936     this.validateStates()
1937     this.validateStates()
1938     this.validateStates()
1939     this.validateStates()
1940     this.validateStates()
1941     this.validateStates()
1942     this.validateStates()
1943     this.validateStates()
1944     this.validateStates()
1945     this.validateStates()
1946     this.validateStates()
1947     this.validateStates()
1948     this.validateStates()
1949     this.validateStates()
1950     this.validateStates()
1951     this.validateStates()
1952     this.validateStates()
1953     this.validateStates()
1954     this.validateStates()
1955     this.validateStates()
1956     this.validateStates()
1957     this.validateStates()
1
```

*Gambar 11: Kode sumber GSP.ts (2)*

```

327 this._state.push(initialConditions);
328 this._stack.push(goalConditions.join(' ^ '));
329 this._stack.push(...goalConditions);
330
331 // Lanjutkan penyelesaian ke iterasi selanjutnya
332 solveNextIteration(options: GSPsolveOptions = {}): GSPIterationReturn {
333   const currentState = this._stack.at(-1);
334   const nextSlot = this._stack.at(-2) ?? '';
335   if (currentState === this._state.at(-1))
336     throw new Error('state kosong, panggil prepare() dahulu!');
337   if (currentState) throw new Error('stack kosong, tidak dapat melanjutkan');
338   const [slotName, A, B] = currentState.match(/w/g) as RegExpMatchArray;
339   const nextSlotMatches = nextSlot.match(/w/g);
340   if (options.draw && options.logging) GSP.printStateObject(this);
341   if (options.logging) this.printCurrentIteration();
342   // Jika slot saat ini terpenuhi oleh state saat ini dan slot dibawahnya
343   // bukan operator, maka pop dari stack
344   // Referensi PDF Planning GSP C1: Langkah 3 kondisi 3
345   if (currentState.includes(currentSlot) && nextSlotMatches) {
346     const nextSlotName = nextSlotMatches[0] as GSPOperator;
347     if (!GSP.OPERATORS.includes(nextSlotName)) return {
348       op: 'pop',
349       object: 'stack',
350       element: this._stack.pop()!
351     };
352   }
353   // Jika state saat ini terdapat slot saat ini, maka pop dari stack
354   // Jika queue teratas tidak terdapat slot saat ini, maka masukkan
355   // Referensi PDF Planning GSP C1: Langkah 3 kondisi 4
356   if (GSP.OPERATORS.includes(slotName as GSPOperator)) {
357     const operation = this._stack.pop() as string;
358     const operator = operation.match(/w/g) as RegExpMatchArray;
359     if (this._queue.at(-1) !== operation) {
360       this._queue.push(operation);
361       this._applyPADI('AD', currentState, operator as GSPOperator, [A, B]);
362     }
363     if (options.draw && options.logging) GSP.printStateObject(this);
364     get currentStateObject() {
365       return {
366         op: 'push',
367         object: 'queue',
368         element: operation
369       };
370     }
371   }
372 }
373
374 // return
375 // Jika slot berisi kondisi atau rangkaian kondisi dan berada di atas slot
376 // yang berisi operator, maka pop slot teratas dari stack dan pop slot
377 // selanjutnya dan masukkan ke queue, dan update current-state
378 // Referensi PDF Planning GSP C1: Langkah 3 kondisi 3
379 if (currentState.includes(nextSlotName)) {
380   const nextSlotMatches = nextSlotMatches[0] as GSPOperator;
381   if (currentSlot.includes(nextSlotName) || slotName === 'HOLDING' && GSP.OPERATORS.includes(nextSlotName)) {
382     const slotConditions = currentSlot.split(' ^ ');
383     const conflict = slotConditions.find((op) => !currentState.includes(op));
384     const popped = this._stack.pop()!;
385     if (conflict) {
386       this._stack.push(conflict);
387       return {
388         op: 'push',
389         object: 'stack',
390         element: conflict
391       };
392     }
393     if (this._stack.length) return {
394       op: 'push',
395       object: 'stack',
396       element: popped
397     };
398     if (GSP.OPERATORS.includes(nextSlotName)) {
399       const operation = this._stack.pop() as string;
400       const [operator, A, B] = operation.match(/w/g) as RegExpMatchArray;
401       this._queue.push(operation);
402       this._applyPADI('AD', currentState, operator as GSPOperator, [A, B]);
403     }
404     if (options.draw && options.logging) GSP.printStateObject(this);
405     get currentStateObject() {
406       return {
407         op: 'push',
408         object: 'stack',
409         element: operation
410       };
411     }
412   }
413   // Jika nama slot adalah suatu kondisi (ON, ONTABLE, ...) maka tambah
414   // operator (STACK, PICKUP, ...) dan PRECONDITION untuk operator tersebut
415   // Referensi PDF Planning GSP C1: Langkah 3 kondisi 2
416   if (GSP.CONDITIONS.includes(slotName as GSPCondition)) {
417     const applyPreconditions = (operator: GSPOperator, blocks: string[]) => {
418       this._applyPADI('P', currentState, operator, blocks);
419       const [newA] = this.getBlockFromStates(currentState, 'ON', [A]);
420       if (newA) {
421         applyPreconditions('UNSTACK', [newA, A]);
422       } else {
423         applyPreconditions('PICKUP', [A]);
424       }
425     };
426     case 'ON': {
427       if (currentState.includes('HOLDING')) {
428         applyPreconditions('PUTDOWN', [A]);
429       } else {
430         applyPreconditions('STACK', [A, B]);
431       }
432     }
433     } break;
434     case 'ONTABLE': {
435       applyPreconditions('PUTDOWN', [A]);
436     }
437     } break;
438     case 'HOLDING': {
439       const [, newB] = this.getBlockFromStates(currentState, 'ON', [A]);
440       if (currentState.includes('ONTABLE({A})')) {
441         applyPreconditions('PICKUP', [A]);
442       } else if (newB) {
443         applyPreconditions('UNSTACK', [A, newB]);
444       } else {
445         applyPreconditions('PUTDOWN', [A]);
446       }
447     }
448     } break;
449     case 'ARMEMPTY': {
450       const [newA] = this.getBlockFromStates(currentState, 'HOLDING', [A]);
451       applyPreconditions('PUTDOWN', [newA]);
452     }
453     } break;
454   }
455   return {
456     op: 'pop',
457     object: 'stack',
458     element: this._stack.pop()!
459   };
460 }
461 }
462
463 solveUntilFinished(options: GSPsolveOptions = {}): void {
464   const colorizer = 'color: white; background-color: green; font-size: 1.05rem;';
465   let iteration = 0;
466   this.prepare();
467   // buat batas iterasi kalau infinit loopop
468   while ((iteration < this.maxIterations) && this._stack.length > 0) {
469     if (options.logging) console.log(`%CLANGKAH KE-${iteration}`, colorizer);
470     this.solveNextIteration(options);
471     if (options.logging) console.log('-----');
472     iteration++;
473   }
474   if (options.logging) console.log('%CHasil iterasi terakhir:', colorizer);
475   if (options.draw) GSP.printStateObject(this.getCurrentStateObject());
476   if (options.logging) this.printCurrentIteration();
477   if (iteration >= this.maxIterations)
478     throw new Error('iterasi lebih dari batas maksimum, memberhentikan!');
479 }
480
481 GSP.ts
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

Gambar 12: Kode sumber GSP.ts (3)

```

328 solveNextIteration(options: GSPsolveOptions = {}): GSPIterationReturn {
329   } break;
330   case 'ARMEMPTY': {
331     const [newA] = this.getBlockFromStates(currentState, 'HOLDING', []);
332     applyPreconditions('PUTDOWN', [newA]);
333   } break;
334 }
335
336 return {
337   op: 'pop',
338   object: 'stack',
339   element: this._stack.pop()!
340 };
341 }
342
343 solveUntilFinished(options: GSPsolveOptions = {}): void {
344   const colorizer = 'color: white; background-color: green; font-size: 1.05rem;';
345   let iteration = 0;
346   this.prepare();
347   // buat batas iterasi kalau infinit loopop
348   while ((iteration < this.maxIterations) && this._stack.length > 0) {
349     if (options.logging) console.log(`%CLANGKAH KE-${iteration}`, colorizer);
350     this.solveNextIteration(options);
351     if (options.logging) console.log('-----');
352     iteration++;
353   }
354   if (options.logging) console.log('%CHasil iterasi terakhir:', colorizer);
355   if (options.draw) GSP.printStateObject(this.getCurrentStateObject());
356   if (options.logging) this.printCurrentIteration();
357   if (iteration >= this.maxIterations)
358     throw new Error('iterasi lebih dari batas maksimum, memberhentikan!');
359 }
360
361 GSP.ts
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

Gambar 13: Kode sumber GSP.ts (4)

```
File Edit View Terminal Tabs Help
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <title>GSP Demo</title>
5 <meta charset="utf-8">
6 <meta name="viewport" content="width=device-width, initial-scale=1">
7 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/tailwindcss/reset/tailwind-compat.min.css">
8 <style[un-cloak]>{display: none;}</style>
9 <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js" defer></script>
10 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.min.js" defer></script>
11 </script>
12 <script>
13 window._doccs = {
14   shortcuts: {
15     'item-container': 'flex flex-col gap-2 min-h-128 overflow-auto',
16     'item-title': 'font-bold text-center text-lg',
17     'state-scene': 'p-1 min-h-16 empty:outline empty:outline-1',
18     'state': 'relative pt-28 w-full flex justify-center items-center outline-1 outline-solid',
19     'arm': 'absolute top-4 mt-8 px-4 py-2 empty:outline-0 outline-1 outline-solid',
20     'table': 'px-8 flex gap-8',
21     'blocks': 'mt-auto',
22     'blocks': 'outline-1 outline-solid px-4 py-2',
23     'button': 'px-4 py-2 rounded text-light',
24     'slot-list': 'p-1 min-h-16 flex flex-col overflow-auto list-decimal list-inside scroll-pa-1',
25     'empty:before:content': '["Currently empty"] before:text-center',
26     'slot': 'p-1 outline outline-1 last:bg-light/15'
27   }
28 </script>
29 </head>
30 <body class="bg-dark text-light text-sm overflow-hidden" un-cloak>
31 <div id="slot-placeholder" class="slot"></div>
32 <div id="state-placeholder" class="state">
33   <p class="absolute px-2 top-2 text-center"></p>
34   <div class="arm"></div>
35   <div class="table"></div>
36 </div>
37 <div class="fixed inset-0 flex flex-col justify-between overflow-hidden">
38   <div class="py-4 bg-dark-200">
39     <div class="font-bold text-base text-center">
40       Block World Problem Goal Stack Planning
41     </div>
42   </div>
43   <main class="p-4 grid grid-cols-1 md:grid-cols-3 grow-1 gap-4 overflow-auto">
44     <section class="item-container md:order-3">
45       <div class="item-title">INITIAL STATE</div>
46       <div id="initial-state" class="state-scene"></div>
47       <div class="item-title">GOAL STATE</div>
48       <div id="goal-state" class="state-scene"></div>
49       <p class="italic">
50         *Block position may be different because the condition doesn't account for position
51       </p>
52     </section>
53     <section class="item-container md:order-2">
54       <div class="item-title">STATE</div>
55       <div id="state-scene" class="state-scene">
56         <div class="state"></div>
57       </div>
58       <div id="state" class="slot-list flex-col-reverse shrink-100"></div>
59     </section>
60     <section class="item-container md:order-1">
61       <div class="item-title">STACK</div>
62       <div id="stack" class="slot-list flex-col-reverse grow-1"></div>
63       <div class="item-title">QUEUE</div>
64       <div id="queue" class="slot-list"></div>
65     </section>
66   </main>
67   <div class="py-4 bg-dark-200 flex flex-wrap justify-center gap-2">
68     <button id="set-states" class="button bg-green-800">Set Initial & Goals</button>
69     <button id="next" class="button bg-blue-800">Next</button>
70     <button id="finish" class="button bg-blue-800">Finish</button>
71     <button id="reset" class="button bg-red-800">Reset</button>
72   </div>
73 </body>
74 </html>
```

Gambar 14: Kode sumber index.html

```
File Edit View Terminal Tabs Help
1 const stateSceneElement = document.querySelector('#state-scene')
2 const stackElement = document.querySelector('#stack')
3 const queueElement = document.querySelector('#queue')
4 const stateElement = document.querySelector('#state')
5
6 const originalStatePlaceholderElement = document.querySelector('#state-placeholder')
7 const statePlaceholderElement = originalStatePlaceholderElement.cloneNode(true)
8
9 statePlaceholderElement.removeAttribute('id')
10 statePlaceholderElement.classList.remove('hidden')
11 originalStatePlaceholderElement.remove()
12
13 const originalSlotElement = document.querySelector('#slot-placeholder')
14 const slotPlaceholderElement = originalSlotElement.cloneNode(true)
15
16 statePlaceholderElement.removeAttribute('id')
17 statePlaceholderElement.classList.remove('hidden')
18 originalSlotElement.remove()
19
20 function createStateElement(stateObject) {
21   const stateElement = statePlaceholderElement.cloneNode(true)
22   const armElement = stateElement.querySelector('.arm')
23   const tableElement = stateElement.querySelector('.table')
24
25   if (stateObject.arm) {
26     armElement.innerText = stateObject.arm
27   } else {
28     armElement.remove()
29   }
30
31   stateElement.querySelector('p').innerText = GSP
32   .generateStateConditions(stateObject)
33   .sort((a, b) => b.localeCompare(a))
34   .join('\n ^ ')
35
36   stateObject.table.forEach((blocks) => {
37     const blocksElement = document.createElement('div')
38     blocksElement.classList.add('blocks')
39     tableElement.insertAdjacentElement('beforeend', blocksElement)
40
41     blocks.forEach((block) => {
42       const blockElement = document.createElement('div')
43       blockElement.innerText = block
44       blockElement.classList.add('block')
45     })
46   })
47   blocksElement.insertAdjacentElement('afterbegin', blockElement)
48 }
49
50 return stateElement
51 }
52
53 const defaultInitial = {
54   arm: null,
55   table: [
56     ['C', 'B', 'A'],
57     ['D']
58   ]
59 }
60
61 const defaultGoal = {
62   arm: null,
63   table: [
64     ['A', 'C'],
65     ['B']
66   ]
67 }
68
69 const gsp = new GSP(defaultInitial, defaultGoal)
70 let currentStatesString = ''
71 let step = 0
72
73 function resetGSP() {
74   step = 0
75   gsp.stack.length = 0
76   gsp.queue.length = 0
77   gsp.state.length = 0
78   stateSceneElement.replaceChildren(statePlaceholderElement)
79   updateStackQueueState()
80   gsp.prepare()
81 }
82
83 function setInitialGoal(initial, goal) {
84   initial ||= defaultInitial
85   goal ||= defaultGoal
86
87   const initialStateElement = createStateElement(initial)
88   const goalStateElement = createStateElement(goal)
89
90   gsp.initial = initial
91   gsp.goal = goal
92 }
93
94 script.js \
95 !LspStop
```

Gambar 15: Kode sumber script.js (1)



```

File Edit View Terminal Tabs Help
93 document.querySelector('#initial-state').replaceChildren(initialStateElement)
1 document.querySelector('#goal-state').replaceChildren(goalStateElement)
2 resetGSP()
3
4
5 function nextIteration() {
6   if (!gsp_stack.length) {
7     updateStackQueueState()
8     setTimeout(() => alert(
9       'Problem has been solved, set another problem or reset current problem'
10      + ' using the buttons below'
11    ), 100)
12   }
13   return
14 }
15 const stateObject = gsp.getCurrentStateObject()
16 const stateElement = createStateElement(stateObject)
17 stateSceneElement.replaceChildren(stateElement)
18 gsp.solveNextIteration()
19 updateStackQueueState()
20 step++
21
22
23
24 function updateStackQueueState() {
25   stackElement.replaceChildren()
26   gsp_stack.forEach((slot) => {
27     const slotElement = slotPlaceholderElement.cloneNode(true)
28     slotElement.innerText = slot
29     stackElement.append(slotElement)
30   })
31   queueElement.replaceChildren()
32   gsp_queue.forEach((slot) => {
33     const slotElement = slotPlaceholderElement.cloneNode(true)
34     slotElement.innerText = slot
35     queueElement.append(slotElement)
36   })
37 }
38
39 stateElement.replaceChildren()
40 gsp_state.forEach((slot) => {
41   const slotElement = slotPlaceholderElement.cloneNode(true)
42   slotElement.innerText = slot.sort((a, b) => b.localeCompare(a)).join(' ^ ')
43   stateElement.append(slotElement)
44 })
script.js \
18 setTimeout(() => stateElement.lastChild.scrollIntoView({ behavior: 'smooth' }), 200)
17
16
15 function setStateInput(name) {
14   const armInput = prompt(`${name} | Arm state (type holding block or leave blank)`)
13   let tableInput = ''
12
11   while (tableInput.length < 1) {
10     tableInput = prompt(
9     `${name} | Table state (Use comma [,] to separate between block stacks)\n`
8     + 'eg. AB, D -> ONTABLE(A) ^ ON(B,A) ^ CLEAR(B) ^ ONTABLE(D)'
7   )
6   if (tableInput.length < 1) alert(`${name} | Table state cannot be empty`)
5
4   const object = {
3     arm: armInput || null,
2     table: []
1
157
1
2   tableInput.trim().split(',').forEach((table, i) => {
3     object.table.push([])
4     table.trim().split('').forEach((block) => object.table[i].push(block))
5   })
6
7   try {
8     GSP.validateStateObject(object)
9   } catch (error) {
10     alert(`Error setting state for ${name}` + error)
11   }
12
13   return object
14 }
15
16 document.addEventListener('DOMContentLoaded', () => {
17   resetGSP()
18   setInitialGoal()
19 })
20
21 document.querySelector('#set-states').addEventListener('click', () => {
22   if (
23     !confirm(
24       'To properly set the states, you have to follow the rules provided,'
25       + ' there will be validation so don\'t worry if you make a mistake'
26     )
27   )

```

Gambar 16: Kode sumber script.js (2)

```

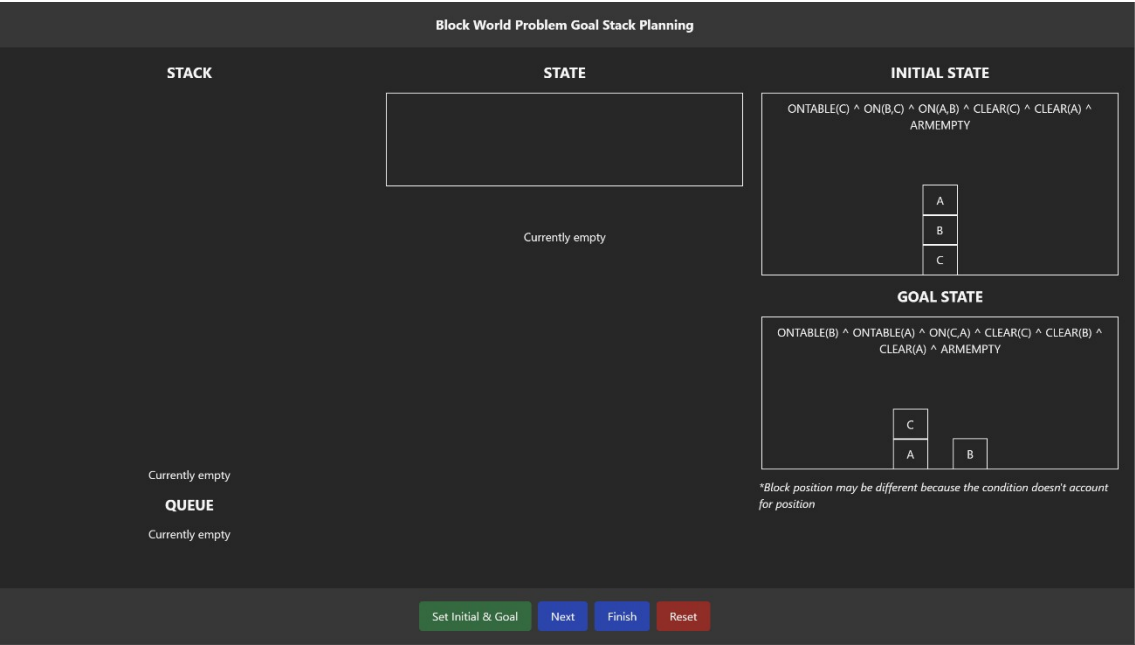
File Edit View Terminal Tabs Help
26 !confirm(
25   'To properly set the states, you have to follow the rules provided,'
24   + ' there will be validation so don\'t worry if you make a mistake'
23 )
22 ) return
21
20 let isError = false
19
18 do {
17   isError = false
16
15   try {
14     const initial = setStateInput('INITIAL')
13     const goal = setStateInput('GOAL')
12     setInitialGoal(initial, goal)
11   } catch (error) {
10     isError = true
9     if (!confirm(`Error setting initial and goal state: ${error}\nRetry?`)) return
8   }
7 } while (isError)
6
5 })
4
3 document.querySelector('#reset').addEventListener('click', resetGSP)
2 document.querySelector('#next').addEventListener('click', nextIteration)
1 document.querySelector('#finish').addEventListener('click', () => {
206   if (!gsp_stack.length) resetGSP()
1   while (gsp_stack.length > 0) nextIteration()
2   nextIteration()
3 })
NORMAL docs script.js \ master // javascript \ UTF-8 206:36 < 98%

```

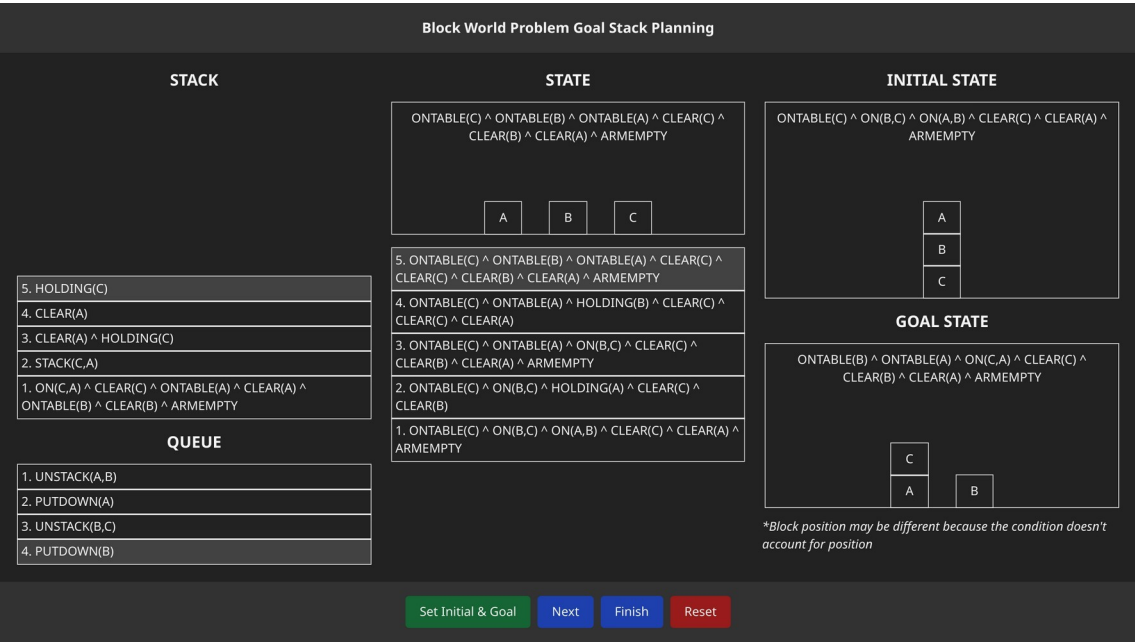
Gambar 17: Kode sumber script.js (3)

4.2. Aplikasi

Implementasi aplikasi untuk proyek laporan ini dapat diakses dengan link <https://sglkc.github.io/goal-stack-planning/>.



Gambar 18: Tangkapan layar implementasi aplikasi



Gambar 19: Tangkapan layar penggunaan aplikasi

## **BAB 5**

### **PENUTUP**

#### **5.1. Kesimpulan**

Goal Stack Planning adalah metode perencanaan yang berguna dalam kecerdasan buatan karena memungkinkan sistem untuk merencanakan tindakan-tindakan yang diperlukan secara hierarkis dan terstruktur untuk mencapai tujuan yang kompleks. Dengan memecah tujuan menjadi langkah-langkah yang lebih kecil dan terkelola, sistem dapat menciptakan rencana yang lebih terorganisasi dan dapat dijalankan dengan lebih efisien.

#### **5.2. Saran**

Aplikasi masih jauh dari kata sempurna dan aplikasi masih mempunyai beberapa kekurangan, dalam implementasi Goal Stack Planning yang diterapkan dalam proyek ini, terdapat beberapa saran yaitu:

1. Kondisi sebuah state sebaiknya mencakup semua posisi block pada meja.
2. Aplikasi membutuhkan perangkat yang memadai untuk memproses setiap langkah dalam waktu yang singkat.

Maka dari itu kami bersedia mendapatkan kritik dan saran dari pembaca untuk lebih membangun hasil yang lebih baik lagi.

## DAFTAR PUSTAKA

- Saraswati, N. W. S. (2015). Optimasi Perencanaan Pengambilan Mata Kuliah Dengan Metode Goal Stack Planning. *S@ CIES*, 5(2), 86-90.
- Nugraheni, C. E., & Abednego, L. (2013). Pemodelan Sudoku sebagai Block World Problem. *Research Report-Engineering Science*, 1.
- Rizaldy, R. E. P., Ahmad, U. A., & Dirgantoro, B. (2023). Implementasi REST API Pada Pengembangan Aplikasi Backend Untuk Platform Kursus Online (Growup). *eProceedings of Engineering*, 10(1).
- Iqbal, M., Husni, M., & Studiawan, H. (2012). Implementasi klien sip berbasis web menggunakan Html5 dan Node. Js. *Jurnal Teknik ITS*, 1(1), A242-A245.
- Siahaan, V., & Sianipar, R. H. (2020). *Buku Pintar JavaScript*. Balige Publishing.
- Novendri, M. S., Saputra, A., & Firman, C. E. (2019). Aplikasi Inventaris Barang Pada Mts Nurul Islam Dumai Menggunakan Php Dan Mysql. *lentera dumai*, 10(2).