Perjanjian No: III/LPPM/2012-09/104-P

PEMODELAN SUDOKU SEBAGAI BLOCK WORLD PROBLEM

Disusun Oleh:
Dr.rer.nat. Cecilia E. Nugraheni
Luciana Abednego, SKom, MT

Lembaga Penelitian dan Pengabdian kepada Masyarakat
Universitas Katolik Parahyangan
2013

Pemodelan Sudoku sebagai Block World Problem

Dr.rer.nat. Cecilia E. Nugraheni,S.T., M.T.,
Luciana Abednego, S.Kom., M.T.

Daftar Isi

Daftar Gambar

Abstrak

Sudoku adalah sejenis teka-teki logika yang tujuan akhirnya adalah mengisikan angka-angka 1 sampai dengan 9 ke dalam suatu kotak berukuran 9×9. Kotak ini memiliki 9 sub-kotak berukuran 3×3. Syarat teka-teki ini adalah tidak ada angka yang berulang pada setiap baris, kolom, atau sub-kotak. Teka-teki Sudoku termasuk ke dalam permasalahan kombinatorial (NP complete). Solusi untuk teka-teki ini dapat dicari dengan bermacam-macam cara seperti algoritma genetik [4], heuristik [1], dan sebagainya. Pada penelitian ini, teka-teki Sudoku akan dicoba dipecahkan dengan memodelkannya sebagai *block-world problem*. Pada *block-world problem*, terdapat sejumlah balok pada meja dengan susunan tertentu. Balok-balok tersebut kemudian diubah susunannya menjadi susunan balok akhir dengan bantuan dua jenis robot. Hasil dari penelitian ini berupa spesifikasi formal dari model Sudoku sebagai *block-world problem* yang ditulis dalam notasi *Temporal Logic of Actions* (TLA).

Bab 1

Pendahuluan

1.1 Latar belakang

Sudoku adalah sejenis teka-teki logika yang tujuan akhirnya adalah mengisikan angka-angka 1 sampai dengan 9 ke dalam suatu kotak berukuran 9×9. Kotak ini memiliki 9 sub-kotak berukuran 3×3. Syarat teka-teki ini adalah tidak ada angka yang berulang pada setiap baris, kolom, atau sub-kotak. Pada saat awal, kotak sudoku belum terisi penuh dengan angka seperti terlihat pada Gambar 1.1. Solusi untuk teka-teki Sudoku sifatnya unik. Solusi untuk teka-teki Sudoku pada Gambar 1.1 dapat dilihat pada Gambar 1.2, ditandai dengan angka berwarna merah. Teka-teki Sudoku pertama kali dipopulerkan sebuah perusahaan teka-teki Jepang bernama Nikoli pada tahun 1986, yang kemudian mendunia pada tahun 2005.

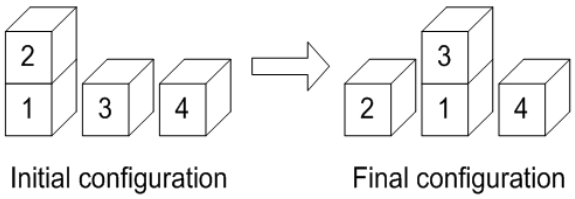
Teka-teki Sudoku termasuk ke dalam permasalahan kombinatorial (*NP complete*). Solusi untuk teka-teki ini dapat dicari dengan bermacam-macam cara seperti algoritma genetik, heuristik, dan sebagainya. Pada penelitian ini akan dilakukan suatu pendekatan yang berbeda untuk pencarian solusi dari permainan Sudoku, yaitu dengan memodelkannya sebagai *block-world problem*. Pada *block-world problem*, terdapat sejumlah balok pada meja dengan susunan tertentu. Balok-balok tersebut kemudian diubah susunannya menjadi susunan balok akhir dengan bantuan dua jenis robot. Robot pertama bertugas memindahkan balok dari tumpukan balok ke meja, sementara robot yang lain bertugas memindahkan balok dari meja ke atas tumpukan balok yang lain. Kedua robot saling bekerja sama dengan berkomunikasi untuk menghasilkan susunan balok akhir. Pada setiap saat, hanya terdapat satu balok yang dapat dipindahkan ke meja atau ke atas tumpukan balok yang lain. Balok yang berada di bawah balok yang lain tidak dapat langsung dipindahkan sebelum balok-balok yang terdapat di atasnya dipindahkan. Gambar

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Gambar 1.1: Contoh teka-teki Sudoku

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Gambar 1.2: Contoh solusi teka-teki Sudoku



Gambar 1.3: Block world problem.

1.3 memperlihatkan contoh tumpukan balok awal dan tumpukan balok akhir. Fokus dari permasalahan *Block-world Problem* adalah pencari solusi yang berupa urutan langkah mulai dari susunan awal sampai dengan susunan akhir.

1.2 Hipotesis

Hipotesis dari penelitian ini adalah permainan Sudoku dapat dipandang sebagai *block-world problem*. Lebih lanjut lagi, permainan Sudoku dapat diselesaikan dengan pendekatan-pendekatan dalam penyelesaian permasalahan *block-world problem*.

1.3 Tujuan penelitian

Untuk itu, permasalahan utama yang diangkat pada penelitian ini bagaimana mencari model yang tepat bagi permainan Sudoku dalam konteks *block-world problem*, hal ini meliputi pemodelan kotak, robot, meja, dan cara kerja dari setiap robotnya.

1.4 Metodologi Penelitian

Metodologi penelitian pada penelitian ini adalah:

- Studi literatur

  Studi literatur dilakukan dengan mempelajari makalah-makalah dari jurnal yang terkait dengan topik penelitian.

- Pemodelan formal

  Dalam pembuatan model nanti, akan digunakan notasi formal yaitu dengan menggunakan notasi Temporal Logic of Actions (TLA).

Bab 2

Studi Pustaka

Studi pustaka yang telah dilakukan sejauh ini adalah dengan mempelajari beberapa hasil penelitian yang dapat ditemukan di internet. Khususnya yang terkait dengan pendekatan/teknik/ algoritma yang digunakan untuk pencarian solusi. Beberapa di antaranya adalah:

1. Pada referensi [1] dibahas pendekatan pencarian solusi permainan Sudoku dengan menggunakan pendekatan pencarian berbasis stokastik, yaitu simulated annealing. Selain itu, pada penelitian ini juga diperkenalkan metode baru untuk membangkitkan papan permainan Sudoku yang dapat diselesaikan dengan simulated annealing.

2. Pada referensi [2] dibahas tentang penyelesaian permainan Sudoku dengan memandang Sudoku sebagai Constraint Satisfaction Problem. Pendekatan yang digunakan adalah constraint programming. Selain masalah pencarian solusi permainan Sudoku, pada makalah ini juga dibahas tentang teknik untuk membangkitkan papan Sudoku dan menentukan tingkat kesulitan dari suatu permainan Sudoku.

3. Pada referensi [3] permainan Sudoku dimodelkan sebagai suatu permasalahan SAT problem. Permainan Sudoku disini direpresentasikan sebagai sekumpulan proposisi dalam bentuk Conjunctive Normal Form (CNF) dari logika proposisi. Pencarian solusinya dilakukan dengan prinsip inferensi dengan menggunakan teknik propagasi unit dan teknik lain yang lebih kompleks.

4. Pada referensi [4] diusulkan pendekatan untuk penyelesaian dan pembangkitan permainan Sudoku dengan evolutionary algorithms, khususnya algoritma genetik. Tujuan dari penelitian yang dilakukan adalah menguji apakah algoritma genetic cukup efisien untuk memecahkan permainan Sudoku dan untuk membangkitkan papan permainan Sudoku.

Sedangkan untuk *Block-world problem*, digunakan sebuah referensi yang merupakan hasil penelitian dari peneliti utama. Pada makalah ini dikaji bagaimana *Block-world Problem* dapat dipandang sebagai suatu*parameterized multi agent system* dan bagaimana permasalahan tersebut dapat dinyatakan secara formal dengan menggunakan notasi Temporal Logic of Actions (TLA) [5].

Bab 3

Dasar Pemodelan

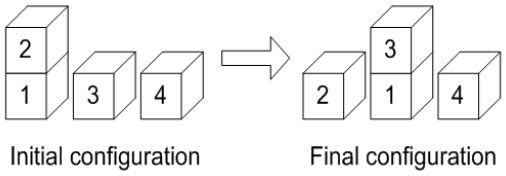
3.1 Model umum untuk parameterized systems

Model yang digunakan digunakan sebagai acuan utama adalah spesifikasi umum untuk *parameterized system*. *Parameterized system* di sini dimengerti sebagai sistem yang terdiri atas sejumlah komponen yang berjalan bersama-sama, banyak komponen tersebut dinyatakan sebagai parameter dari sistem. Pada penelitian ini *parameterized system* dibatasi pada sistem yang bersifat *interleaving* dimana setiap saat hanya ada satu komponen yang aktif atau melakukan aksi. Spesifikasi dari sistem ini berbentuk sbb.:

$$parSpec \equiv \forall k \in M \ : \ Init(k) \wedge \Box[Next(k)]_{v[k]} \wedge L(k).$$

dengan

- *Init* adalah predikat status yang menjelaskan kondisi awal secara global,
- $Next(k)$ adalah sebuah aksi (fungsi transisi) yang menyatakan relasi *next-state* dari suatu proses k,
- v adalah suatu fungsi status yang menyatakan variabel dari sistem, dan
- $L(k)$ adalah suatu formula yang menyatakan kondisi *liveness* yang diharapkan dari proses dari subsistem k.



Gambar 3.1: Contoh *block world problem.*

3.2 Model dari block-world problem

3.2.1 Deskripsi masalah

Block world problem secara ringkas dapat dijelaskan sebagai berikut [**?**]: diberikan sekumpulan balok di atas meja dan dua tipe robot yang bertugas untuk mengubah tumpukan balok, dari konfigurasi awal menjadi konfigurasi akhir. Diasumsikan bahwa hanya satu balok yang dapat dipindahkan setiap saat: apakah ditempatkan di atas meja atau di atas balok yang lain. Sebarang balok yang berada di bawah balok yang lain tidak dapat dipindahkan. Setiap robot mempunyai kemampuan yang berbeda: robot pertama hanya dapat memindahkan balok dari atas tumpukan ke meja, sementara robot kedua hanya dapat memintahkan balok dari atas meja ke atas tumpukan balok.

Sebagai ilustrasi, Gambar 3.1 menunjukkan sebuah contoh dari permasalahan ini. Terdapat empat balok di atas meja. Setiap balok diberi nomor. Sebelah kiri menyatakan kondisi awal dan sebelah kanan kondisi akhir yang diinginkan.

3.2.2 Spesifikasi

Dari deskripsi permasalahan, jelas bahwa agen untuk *block-world problem* tidak homogen. Meskipun demikian permasalahan ini tetap dapat diperlakukan sebagai *parameterized system* dan dinyatakan dengan formula umum untuk *parameterized system.* Aksi untuk setiap agen seragam, hanya pada aksi tersebut diberikan prekondisi tertentu, sehingga agen-agen hanya dapat menjalan aksi yang benar-benar menjadi tugasnya.

Spesifikasi dari *block-world problem* diberikan pada Gambar 3.2. Struktur data dan variabel yang digunakan pada spesifikasi ini adalah:

- *Block* adalah sebuah himpunan bilangan bulat untuk merepresentasikan kumpulan kotak.
- *ag_type* sebuah array satu dimensi untuk mengidentifikasikan tipe agen.

- Untuk merepresentasikan status atau konfigurasi kotak digunakan array yang elemennya berupa pasangan bilangan bulat tak negatif. Setiap element ini menyatakan kondisi suatu kotak. Sebagai contoh, jika elemen kedua dari array adalah $\langle 3, 1\rangle$ maka berarti kotak nomor 3 berada di bawah kotak nomor 2 dan kota nomor 1 berada di atas kotak nomor 3. Angka khusus, 0, digunakan untuk menyatakan meja atau tidak ada apa-apa.

  Konfigurasi awal untuk Gambar 3.1 dinyatakan sebagai $\langle\langle 0, 2\rangle, \langle\langle 1, 0\rangle, \langle 0, 0\rangle, \langle 0, 0\rangle\rangle$ dan konfigurasi akhir dinyatakan sebagai $\langle\langle 0, 3\rangle, \langle 0, 0\rangle, \langle 1, 0\rangle, \langle 0, 0\rangle\rangle$.

- Tiga status array *IState*, *CState* dan *FState*, masing-masing digunakan untuk menyatakan konfigurasi awal, akhir, dan yang sedang berlaku (*current*).

Sedangkan aksi yang digunakan adalah:

- Aksi *move*(k) hanya dapat diambil jika agen dengan kemampuan memindahkan kotak dari meja dan meletakkannya di atas tumpukan kotak yang lain.

- Aksi *free*(k) hanya dapat diambil oleh agen yang kemampuannya adalah mengambil kotak dari suatu tumpukan kotak dan memindahkannya ke atas meja.

Nilai dari *Blocks*, *ag_type*, *IState*, dan *FState* bergantung pada instansi permasalahan sedang dihadapi. Jika jumlah kotak dan tiep agen berubah, maka diperlukan sedikit modifikasi pada *Blocks* dan *ag_types* agar sesuai dengan masalah.

module *Block-world problem*

$$
\begin{aligned}
isDone &\equiv noMoreNumber \lor \ noMoreMove \\
move(k) &\equiv \land \ \neg isDone \land ag\_type[k] = 1 \\
&\quad \land \ \exists x, y \in Blocks \ : \\
&\quad \land \ x \neq y \\
&\quad \land CState[x] = \langle 0, 0 \rangle \land CState[y][2] = 0 \\
&\quad \land CState' = [CState \ EXCEPT \, ![x][1] = y, \\
&\qquad\qquad\qquad\qquad ![y][2] = x] \\
free(k) &\equiv \land \ \neg isDone \land ag\_type[k] = 2 \\
&\quad \land \ \exists x, y \in Blocks \ : \\
&\quad \land \ x \neq y \\
&\quad \land CState[x][2] = 0 \land CState[y][2] = x \\
&\quad \land CState' = [CState \ EXCEPT \, ![x] = \langle 0, 0 \rangle, \\
&\qquad\qquad\qquad\qquad ![y][2] = 0] \\
Init &\equiv \land \ CState = IState \land CState \neq FState \\
&\quad \land \ \forall k \in M \ : \ ag\_type[k] \in \{1, 2\} \\
Next(k) &\equiv Move(k) \lor Free(k) \\
L(k) &\equiv WF_v(Move(k)) \land WF_v(Free(k)) \\
v &\equiv \langle CState \rangle \\
BWorld &\equiv \land Init \land \Box[\exists k \in M \ : \ Next(k)]_v \\
&\quad \land \forall k \in M \ : \ L(k)
\end{aligned}
$$

Gambar 3.2: Spesifikasi untuk *Block World Problem*.

Bab 4

Model Sudoku

4.1 Deskripsi informal

Untuk dapat memodelkan sudoku sebagai *block-world problem*, maka perlu melakukan analisis terhadap komponen dari masing-masing permasalahan:

- Komponen dari *block-world problem* adalah sekumpulan robot dan lingkungan yang terdiri atas kotak-kotak dan meja. Kotak-kotak tersebut tersusun sedemikian rupa di atas meja, kemudian robot secara bergantian akan memindahkan kotak-kotak tersebut sampai tercapai susunan yang diinginkan.
- Sedangkan komponen dari dari sudoku adalah papan sudoku dan angka-angka. Pada saat awal, sebagian angka sudah berada di papan, dan sebagian lainnya berada di luar papan. Hal ini berbeda dengan *block-world problem* dimana tidak ada papan khusus untuk menempatkan balok-balok.

Dari pengamatan tersebut maka pada penelitian ini Sudoku dipandang sebagai varian dari *block-world problem* dengan lingkungan berupa papan berupa *array* dua dimensi berukuran 9×9 dan tumpukan kotak-kotak bernomor dari 1 s/d 9 dengan masing-masing nomor berjumlah 9 buah. Papan sudoku dapat juga dipandang sebagai matriks berukuran 3×3 dengan setiap elemen matriks berupa subblok yaitu matriks berukuran 3×3.

Pada saat awal sebagian kotak-kotak tersebut sudah berada atau diletakkan pada papan dan kotak-kotak lainnya berada di luar papan. Kotak yang berada di papan tidak boleh ditumpuk dengan kotak yang lain dan harus memenuhi aturan Sudoku yaitu tidak boleh ada kotak dengan nomor yang sama dalam satu baris, satu kolom, maupun satu subblok (matriks berukuran

3×3). Tujuan dari Sudoku ini mencari konfigurasi akhir yaitu pengaturan kotak sedemikian rupa sehingga seluruh kotak berada di papan dan memenuhi aturan Sudoku yang berlaku.

Mirip pada *block-world problem*, kemampuan robot pada Sudoku ada dua yaitu:

1. tipe 1: mengambil kotak yang berada di luar papan dan meletakkannya ke papan
2. tipe 2: mengambil kotak dari papan dan mengembalikannya ke tumpukan di luar papan

Model yang sudah berhasil dikembangkan ada tiga. Diasumsikan, untuk ketiga model, kotak-kotak tersusun kedalam sembilan tumpukan sesuai dengan nomor kotak yaitu 1 s/d 9. Untuk model pertama, digunakan dua buah robot yang masing-masing fungsinya sesuai dengan kemampuan robot yang sudah dijelaskan sebelumnya. Sedang untuk model kedua dan ketiga digunakan sembilan buah robot yang bertipe sama, yaitu robot jenis 1. Masing-masing robot ini bertanggung jawab untuk meletakkan kotak dengan nomor tertentu ke papan. Pada model kedua dan ketiga kotak diletakkan hanya jika robot telah menemukan lokasi yang tepat bagi kotak tersebut. Sebagai akibatnya tidak diperlukan robot dengan kemampuan mengambil kotak dari papan dan mengembalikan ke tumpukan.

Jika dilihat dari *parameterized system*, dapat disimpulkan bahwa model 1 termasuk dalam *heterogen parameterized system*, sementara model 2 dan model 3 termasuk dalam *homogen parameterized system*. Spesifikasi formal dari ketiga model ini mengikuti spesifikasi umum dari *parameterized system*, dan khususnya mengikuti spesifikasi dari *block-world problem*. Pada spesifikasi *block-world problem* aksi dari setiap agen terdiri atas dua yaitu *move* dan *free* yang sebenarnya aksi itu menyatakan kemampuan dari masing-masing robot.

4.1.1 Model 1

Robot pertama bertugas untuk meletakkan kotak ke papan. Hal ini dilakukan dengan cara mencari lokasi di papan yang masih kosong dan mencari kotak yang dapat diletakkan pada lokasi tersebut. Pencarian selalu dimulai dari lokasi terdepan dan pencarian kotak akan dimulai dari kotak dengan nomor terkecil. Jika suatu saat robot satu gagal menemukan kotak yang tepat untuk suatu lokasi, maka akan memberitahu robot kedua bahwa telah terjadi kegagalan. Robot kedua akan mengambil kotak terakhir yang diletakkan oleh robot satu dan mengembalikannya ke tumpukan kotak yang sesuai. Robot satu kemudian akan mencoba menemukan kotak lain yang cocok untuk lokasi tersebut.

Di sini perlu terjalin komunikasi antara robot satu dan robot kedua, khususnya pada saat terjadi kegagalan. Untuk tujuan tersebut diasumsikan ada sebuah lampu indikator di lingkungan. Pada saat awal lampu ini dalam kondisi mati. Jika pada suatu saat robot pertama gagal

menemukan posisi untuk menempatkan kotak di papan, maka robot akan menyalakan lampu ini. Robot kedua, hanya bekerja jika lampu ini menyala, dia akan mengambil kotak terakhir yang diletakkan oleh robot satu dan mengembalikannya ke tumpukan dan mematikan lampu tersebut.

4.1.2 Model 2

Pada model 2 digunakan sembilan robot seragam bertipe satu, yaitu robot yang bertugas untuk mengambil sebuah kotak dengan nomor tertentu sesuai dengan tanggungjawabnya dan meletakkannya ke papan sudoku. Pencarian solusi dilakukan dengan cara mengulang putaran giliran robot sampai kondisi berhenti tercapai.

Robot secara bergilir, dimulai dari robot 1 sampai dengan robot 9, mencari sebuah posisi di papan yang dapat ditempati kotak dengan angka tertentu. Jika posisi ditemukan, robot tersebut akan meletakkan kotak ke posisi yang tepat. Jika tidak ditemukan, robot tidak melakukan apa-apa dan giliran diberikan kepada robot selanjutnya. Setelah robot 9 mendapat giliran, maka akan dilihat apakah dalam satu putaran tersebut ada kotak yang dapat diletakkan atau apakah ada robot yang berhasil melaksanakan tugasnya. Jika ada, maka pencarian solusi diteruskan dengan membuat putaran baru. Jika dalam suatu putaran tidak ada robot yang berhasil maka pencarian solusi dihentikan.

Pada model 2 ini yang dimaksud dengan posisi yang tepat dapat dijelaskan sebagai berikut. Sebuah robot ke-i akan menyebut sebuah posisi dengan baris x dan kolom y tepat jika kondisi ini terpenuhi:

- belum ada kotak di posisi tersebut
- tidak ada posisi lain pada baris x yang berisi kotak dengan nomor i
- tidak ada posisi lain pada kolom y yang berisi kotak dengan nomor i
- tidak ada posisi lain pada subblok yang sama yang berisi kotak dengan nomor i
- pada dua baris lain dalam subblok yang sama terdapat posisi berisi kotak dengan nomor i,
- pada dua kolom lain dalam subblok yang sama terdapat posisi berisi kotak dengan nomor i

4.1.3 Model 3

Pada dasarnya model 3 ini mirip dengan model 2, bedanya adalah pada penentukan suatu posisi apakah tepat atau tidak. Pada model 3 yang dimaksud dengan posisi yang tepat dapat dijelaskan sebagai berikut. Sebuah robot ke-i akan menyebut sebuah posisi dengan baris x dan kolom y tepat jika kondisi ini terpenuhi:

- belum ada kotak di posisi tersebut
- tidak ada posisi lain pada baris x yang berisi kotak dengan nomor i
- tidak ada posisi lain pada kolom y yang berisi kotak dengan nomor i
- tidak ada posisi lain pada subblok yang sama yang berisi kotak dengan nomor i
- delapan posisi lain yang merupakan tetangga satu subblok memenuhi kondisi:
  - belum ada kotak di posisi tersebut, atau
  - dijamin bahwa posisi tersebut tidak akan dapat diisi dengan kotak dengan nomor i dengan melihat apakah ada kotak dengan nomor i pada baris x atau kolom y di subblok yang lain

4.2 Spesifikasi formal

4.2.1 Struktur data dan variabel sistem

Variabel sistem untuk Sudoku adalah sbb.:

1. *Board*

   Board merepresentasikan papan Sudoku, berupa sebuah *array* dengan masing-masing elemennya adalah tripel yaitu tiga buah bilangan bulat. Tiga bilangan bulat ini menyatakan baris, kolom, dan nilai kotak yang ditempatkan pada lokasi tersebut. Contoh $\langle 3, 4, 2 \rangle$ menyatakan bahwa baris ke-3 dan kolom ke-4 berisi kotak dengan nomor 2. Jika suatu lokasi belum pernah terisi kotak, maka nilai elemen ke-3 adalah 0.

   Sebagai catatan, indeks dimulai dari angka 0, sehingga indeks elemen pertama dari matriks *Board* adalah *Board*[0][0].

2. *numbers*

numbers merepresentasikan jumlah tumpukan kotak yang belum diletakkan di papan Sudoku. *numbers* berupa sebuah *array* dengan masing-masing elemen adalah pasangan terurut dua buah bilangan bulat, yang menyatakan nomor kotak dan jumlah kotak yang belum diletakkan di papan Sudoku untuk nomor tersebut.

3. *FC*

   FC adalah daftar sel yang masih kosong (*free cell*). Daftar sel ini diorganisasikan sebagai array yang setiap elemennya berupa tripel dengan masing-masing nilai berupa bilangan bulat. Tiga nilai tersebut, sama seperti elemen *Board* menyatakan baris, kolom, dan nomor kotak yang ditempatkan pada posisi tersebut.

4. *First*

   First adalah bilangan bulat tak negatif yang menyatakan indeks elemen pertama dari *FC*. *First* digunakan pada model 1.

5. *Last*

   Last adalah bilangan bulat tak negatif yang menyatakan indeks elemen terakhir dari *FC*. *Last* digunakan pada model 1.

6. *isBackTrack*

   isBackTrack adalah sebuah variabel boolean yang digunakan untuk mengindikasi apakah terjadi *backtrack* atau tidak. Variabel ini hanya digunakan pada model 1.

7. *turn*

   turn adalah variabel integer yang digunakan untuk mengatur giliran robot pada model 2 dan 3.

8. *success*

   success adalah variabel boolean yang digunakan untuk mengetahui apakah ada robot yang berhasil meletakkan sebuah kotak pada papan dalam satu iterasi tertentu. Variabel ini digunakan pada model 2 dan 3.

9. *isDone*

   isDone adalah variabel bertipe boolean yang digunakan sebagai indikator apakah proses pencarian solusi masih dilanjutkan atau tidak. Variabel ini digunakan pada model 2 dan 3. Sedangkan pada model 1, *isDone* digunakan bukan sebagai variabel tetapi sebagai nama ekspresi/fungsi.

9								8
	3			8			6	
		8		2		1		
			8	5	3			
	4	6	7		2	8	2	
			3	4	5			
		1				3		
	8						9	
5								2

Gambar 4.1: Contoh Soal Sudoku.

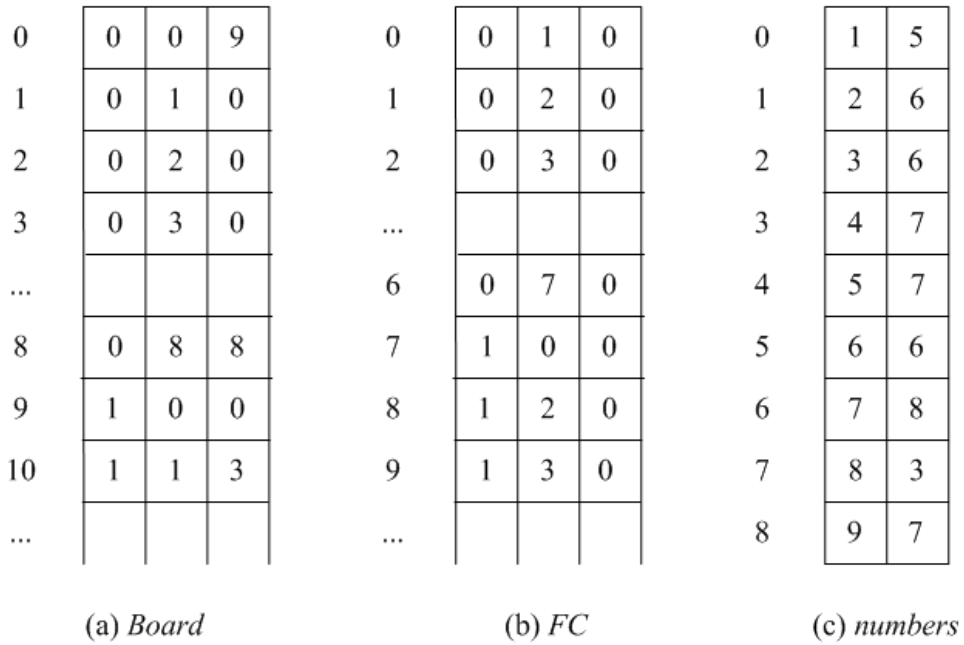
Untuk memperjelas struktur data dan variabel sistem, dimisalkan soal Sudoku yang dipecahkan adalah seperti pada Gambar 4.1. Struktur data dan nilai dari masing-masing variabel *Board*, *FC*, dan *numbers* ditunjukkan pada Gambar 4.2.

Spesifikasi umum dari ketiga model diberikan pada Gambar 4.3. Pada spesifikasi tersebut beberapa variabel sistem tidak diberikan secara eksplisit pada saat awal, yaitu *Board*, *FC*, *numbers*, dan *Last*. Nilai awal dari ketiga variabel tersebut tergantung dari permasalahan Sudoku yang sedang ditangani. Diasumsikan bahwa pada saat awal keempat variabel tersebut sudah terisi dengan nilai yang sesuai. Aksi $PutOn(k)$ dan $TakeBack(k)$ juga belum didefinisikan pada spesifikasi tersebut. Aksi ini akan didefinisikan lebih lanjut pada pembahasan masing-masing model.

4.2.2 Model 1

Spesifikasi untuk model 1 diberikan pada Gambar 4.4 dan Gambar 4.5. Pada model 1 nilai M yang menyatakan jumlah/jenis robot adalah 2. Variabel sistem yang digunakan adalah $Board, FC, numbers, First, Last$, dan $isBackTrack$.

Selain spesifikasi dari aksi $PutOn(k)$ dan $TakeBack(k)$ didefinisikan juga beberapa aksi/fungsi bantuan yang digunakan pada spesifikasi, yaitu:



0	0	0	9
1	0	1	0
2	0	2	0
3	0	3	0
...			
8	0	8	8
9	1	0	0
10	1	1	3
...			

(a) *Board*

0	0	1	0
1	0	2	0
2	0	3	0
...			
6	0	7	0
7	1	0	0
8	1	2	0
9	1	3	0
...			

(b) *FC*

0	1	5
1	2	6
2	3	6
3	4	7
4	5	7
5	6	6
6	7	8
7	8	3
8	9	7

(c) *numbers*

Gambar 4.2: Struktur data dan isi variabel sistem.

module *Sudoku*

$PutOn(k) \triangleq ...$

$TakeBack(k) \triangleq ...$

$Init \triangleq \wedge\ Board = ...$

$\quad \wedge\ FC = ...$

$\quad \wedge\ numbers = ...$

$\quad \wedge\ ...$

$Next(k) \triangleq PutOn(k) \vee TakeBack(k)$

$L(k) \triangleq WF_v(PutOn(k)) \wedge WF_v(TakeBack(k))$

$v \triangleq \langle Board, FC, numbers, ... \rangle$

$Sudoku \triangleq \wedge\ Init$

$\quad \wedge\ \Box[\exists k \in M\ :\ Next(k)]_v$

$\quad \wedge\ \forall k \in M\ :\ L(k)$

Gambar 4.3: Spesifikasi Sudoku secara umum.

1. *isDone*

   isDone adalah sebuah ekspresi yang digunakan untuk menguji apakah proses pencarian solusi akan dilanjutkan atau tidak. Pengujian dilakukan dengan menguji kebenaran ekspresi $\forall i \in \langle 0..8 \rangle : numbers[i][2] = 0$.

2. *isNeighbor*(x, y)

   isNeighbor adalah fungsi untuk menguji apakah baris atau kolom x berada pada subblok yang sama dengan baris atau kolom y. Pengujian ini dilakukan dengan menguji kebenaran ekspresi $\exists n : r \ div \ 3 = n \wedge x \ div \ 3 = n$.

3. *isEmpty*(x, y)

   isEmpty adalah fungsi untuk menguji apakah posisi pada baris x dan kolom y sudah terisi kotak atau belum. Pengujian ini dilakukan dengan menguji kebenaran ekspresi $\exists i : Board[i] = \langle x, y, 0 \rangle$ dimana 0 menyatakan kalau posisi tersebut masih kosong.

4. *isInRow*(x, i)

   isInRow(x, i) adalah fungsi untuk menguji apakah pada baris x sudah ada posisi yang berisi kotak bernomor i. Pengujian ini dilakukan dengan menguji kebenaran ekspresi $\exists j, y : Board[j] = \langle x, y, i \rangle$.

5. *isInColumn*(y, i)

   isInColumn(y, i) adalah fungsi untuk menguji apakah pada kolom y sudah ada posisi yang berisi kotak bernomor i. Pengujian ini dilakukan dengan menguji kebenaran ekspresi $\exists j, x : Board[j] = \langle x, y, i \rangle$.

6. *isInSubBlock*(x, y, i)

   isInSubBlock(x, y, i) adalah fungsi untuk menguji apakah pada subblock dimana posisi (x, y) berada sudah berisi kotak bernomor i.

7. *isValidPosition*(x, y, i)

   isValidPosition(x, y, i) adalah fungsi untuk menguji apakah kotak bernomr i dapat diletakkan pada posisi baris x dan kolom y. Fungsi ini akan mengembalikan nilai TRUE jika ekspresi $\neg isEmpty(x, y) \wedge \neg isInRow(x, i) \wedge \neg isInColumn(y, i) \wedge \neg inInSubBlock(x, y, i)$.

module *Model 1*

$isDone \triangleq \forall i \in \langle 0..8 \rangle \ : \ numbers[i][2] = 0$

$isNeighbor(r, x) \triangleq \exists n \ : \ r \ div \ 3 = n \land x \ div \ 3 = n$

$isEmpty(x, y) \triangleq \exists i \ : \ Board[i] = \langle x, y, 0 \rangle$

$isInRow(x, i) \triangleq \exists j, y \ : \ Board[j] = \langle x, y, i \rangle$

$isInColumn(y, i) \triangleq \exists j, x \ : \ Board[j] = \langle x, y, i \rangle$

$isInSubBlock(x, y, i) \triangleq \exists j, k \ : \ \land \ \langle x, y \rangle \neq \langle j, k \rangle$
$\land \ isNeighbor(x, j)$
$\land \ isNeighbor(y, k)$
$\land \ \exists l \ : \ Board[l] = \langle j, k, i \rangle$

$isValidPosition(x, y, i) \triangleq \land \ isEmpty(x, y, i)$
$\land \ \neg isInRow(x, i)$
$\land \ \neg isInColumn(y, i)$
$\land \ \neg isInSubBlock(x, y, i)$

$PutOn(k) \triangleq$

$\land \ \neg isDone \ \land k = 1$

$\land$ IF $\land \ \exists i \ : \ i \geq First \land i \leq Last \land isValidPosition(FC[First][1], FC[First][2], i)$
$\land \ \exists j \ : \ Board[j] = \langle FC[First][1], FC[First][2], 0 \rangle$
THEN $\land \ Board' = [Board$ EXCEPT $![j] = \langle FC[First][1], FC[First][2], i \rangle$
$\land \ FC' = [FC$ EXCEPT $![First][3] = i]$
$\land \ isBackTrack' =$ FALSE
$\land \ First' = First + 1$
$\land$ UNCHANGED $\langle numbers, Last \rangle$
ELSE IF $isBackTrack$
THEN $\land \ FC' = [FC$ EXCEPT $![First][3] = 0]$
$\land$ UNCHANGED $\langle Board, isBackTrack, First, Last, isDone, Lastnumbers \rangle$
ELSE $isBackTrack' =$ TRUE
UNCHANGED $\langle Board, FC, First, Last, idDone, numbers \rangle$

Gambar 4.4: Spesifikasi Sudoku Model 1.

module *Model 1 - continued*

$$
\begin{array}{l}
TakeBack(k) \triangleq \\
\land\ \neg isDone\ \land k = 2\ \land isBacktrack \\
\land\ \text{IF}\ First = 0\ \text{THEN}\ \land\ isBackTrack' = \text{FALSE} \\
\qquad\qquad\qquad\qquad \land\ \text{UNCHANGED}\ \langle Board, FC, First, Last, numbers \rangle \\
\qquad\qquad\quad \text{ELSE}\ \text{LET}\ x = FC[First - 1][1] \\
\qquad\qquad\qquad\qquad\ \ y = FC[First - 1][2] \\
\qquad\qquad\qquad\quad \text{IN}\ \ \exists i\ :\ \land\ Board[i][1] = x \land Board[i][2] = y \\
\qquad\qquad\qquad\qquad\qquad\ \land\ FC' = [FC\ \text{EXCEPT}\ ![First - 1][3]' = Board[i][3]] \\
\qquad\qquad\qquad\qquad\qquad\ \land\ numbers' = [numbers\ \text{EXCEPT}\ ![Board[i][3]] = numbers[Board[i][3]] + 1] \\
\qquad\qquad\qquad\qquad\qquad\ \land\ Board' = [Board\ \text{EXCEPT}\ ![i][3] = 0] \\
\qquad\qquad\qquad\qquad\qquad\ \land\ First' = First - 1 \\
\qquad\qquad\qquad\qquad\qquad\ \land\ \text{UNCHANGED}\ \langle Last, isBackTrack \rangle \\
Next(k) \triangleq PutOn(k) \lor TakeBack(k) \\
L(k) \triangleq WF_v(PutOn(k)) \land WF_v(TakeBack(k)) \\
v \triangleq \langle Board, FC, numbers, ... \rangle \\
Sudoku \triangleq \land\ Init \\
\qquad\qquad \land\ \Box[\exists k \in M\ :\ Next(k)]_v \\
\qquad\qquad \land\ \forall k \in M\ :\ L(k)
\end{array}
$$

Gambar 4.5: Spesifikasi Sudoku Model 1 (lanjutan).

4.2.3 Model 2

Spesifikasi untuk model 2 diberikan pada Gambar 4.6. Nilai M pada model ini adalah 9. Variabel sistem yang digunakan pada spesifikasi ini adalah $Board, FC, numbers, turn$, dan $success$. Pada spesifikasi model 2 ini digunakan juga beberapa aksi/fungsi tambahan seperti pada model 1, yaitu: $isEmpty(x, y, i)$, $isInRow(x, i)$, $isInColumn(y, i)$, dan $isInSubBlock(x, y, i)$. Fungsi lainnya yang berbeda dengan model 1 adalah sbb.:

1. $CheckOtherRows(x, i)$

   $CheckOtherRows(x, i)$ adalah fungsi untuk menguji kemunculan kotak bernomor i pada dua baris lain yang berada dalam satu subblok yang sama dengan baris x.

2. $CheckOtherColumn(y, i)$

   $CheckOtherColumn(y, i)$ adalah fungsi untuk menguji kemunculan kotak bernomor i pada dua kolom lain yang berada dalam satu subblok yang sama dengan kolom y.

3. $isValidPosition(x, y, i)$

   $isValidPosition(x, y, i)$ adalah fungsi untuk menguji apakah posisi (x, y) adalah posisi yang tepat bagi kotak bernomor i berdasarkan aturan model 2.

Pada model 2 ini, aksi $TakeBack(k)$ menjadi semacam aksi *dummy*. Aksi ini senantiasa aktif tetapi pengaktifan aksi ini tidak memberikan dampak perubahan pada variabel sistem. Walaupun sebenarnya, aksi bisa saja dihilangkan dari spesifikasi, aksi tetap dipertahankan untuk menjaga keseragaman dengan spesifikasi umum yang diberikan pada Gambar 4.3.

Cara kerja dari robot pada model dua ini menggunakan prinsip *fixed-point* yaitu mengulang proses pencarian posisi yang tepat bagi kotak-kotak oleh robot secara bergiliran mulai dari robot ke-1 sampai dengan ke-9 sampai tidak terjadi perubahan terhadap variabel sistem. Jika dalam suatu iterasi tidak ada robot yang berhasil meletakkan kotak, maka proses dihentikan. Untuk mengetahui apakah iterasi dilanjutkan atau tidak, digunakan variabel *success* yang akan bernilai FALSE jika tidak ada satu robot pun yang berhasil meletakkan kotak yang menjadi tugasnya.

module *Model 2*

$$isNeighbor(r, x) \triangleq \exists n \,:\, r \;div\; 3 = n \land x \;div\; 3 = n$$

$$\begin{aligned} CheckOtherRows(r_1, r_2, x, i) \triangleq \land\ & r_1 \neq r_2 \land r_1 \neq x \land r_2 \neq x \\ & \land\ isNeighbor(r_1, x) \land isInRow(r_1, i) \\ & \land\ isNeighbor(r_2, x) \land isInRow(r_2, i) \end{aligned}$$

$$\begin{aligned} CheckOtherColumns(c_1, c_2, y, i) \triangleq \land\ & c_1 \neq c_2 \land c_1 \neq y \land c_2 \neq y \\ & \land\ isNeighbor(c_1, y) \land isInColumn(c_1, i) \\ & \land\ isNeighbor(c_2, y) \land isInColumn(c_2, i) \end{aligned}$$

$$\begin{aligned} isValidPosition(x, y, i) \triangleq \land\ & isEmpty(x, y) \\ & \land \neg isInRow(x, i) \land \neg isInColumn(y, i) \land \neg isInSubBlock(x, y, i) \\ & \land \exists r_1, r_2 \,:\, CheckOtherRows(r_1, r_2, x, i) \\ & \land \exists c_1, c_2 \,:\, CheckOtherColumns(c_1, c_2, y, i) \end{aligned}$$

$$PutOn(k) \triangleq$$

$$\land \neg isDone \;\land k = turn$$

$$\land \text{ IF } \exists i \,:\, FC[i][3] = 0 \land\ isValidPosition(FC[i][1], FC[i][2], k)$$

$$\text{THEN } \exists j \,:\, \land\ Board[j] = \langle FC[i][1], FC[i][2], 0 \rangle$$

$$\land\ Board' = [Board \text{ EXCEPT } ![j][3] = k]$$

$$\land\ FC' = [FC \text{ EXCEPT } ![i][3] = k]$$

$$\land\ numbers' = [numbers \text{ EXCEPT } ![Board[j][3]] = numbers[Board[j][3]] - 1]$$

$$\land\ success' = \text{TRUE}$$

$$\text{ELSE UNCHANGED } \langle Board, FC, numbers \rangle$$

$$\begin{aligned} \land \text{ IF } turn < 9 \text{ THEN } & \land\ turn' = turn + 1 \\ & \land \text{ UNCHANGED } \langle success, isDone \rangle \\ \text{ELSE } & \land \text{ IF } \neg success \text{ THEN } \land\ isDone' = \text{TRUE} \\ & \qquad\qquad\qquad\qquad \land \text{ UNCHANGED } \langle success, turn \rangle \\ & \text{ELSE } \land\ turn' = 1 \\ & \qquad\qquad\qquad\qquad \land\ success' = \text{FALSE} \\ & \qquad\qquad\qquad\qquad \land \text{ UNCHANGED } isDone \end{aligned}$$

$$TakeBack(k) \triangleq \text{UNCHANGED } v$$

$$v \triangleq \langle Board, FC, numbers, turn, success, isDone \rangle$$

Gambar 4.6: Spesifikasi Sudoku Model 2.

module *Model 2 - continued*

$Init \triangleq \land\ turn = 1 \land isDone = \text{FALSE} \land success = \text{FALSE}$ $Next(k) \triangleq PutOn(k) \lor TakeBack(k)$

$\land\ Board = ...$

$\land\ FC = ...$

$\land\ numbers = ...$

$L(k) \triangleq WF_v(PutOn(k)) \land WF_v(TakeBack(k))$

$v \triangleq \langle Board, FC, numbers, isDone, turn, success \rangle$

$Sudoku2 \triangleq \land\ Init$

$\land\ \Box[\exists k \in M\ :\ Next(k)]_v$

$\land\ \forall k \in M\ :\ L(k)$

Gambar 4.7: Spesifikasi Sudoku Model 2 (lanjutan).

4.2.4 Model 3

Spesifikasi untuk model 3 diberikan pada Gambar 4.8 dan 4.9. Spesifikasi ini pada prinsipnya sangat mirip dengan spesifikasi model 2. Perbedaan dengan spesifikasi model 2 adalah pada definisi dari fungsi $isValidPosition(x, y, i)$ yang digunakan untuk menguji apakah suatu posisi (x, y) pada *Board* dapat diisi dengan kotak bernomor i. Fungsi ini menggunakan fungsi bantuan $CheckNeighborCells(x, y, i)$ yang digunakan untuk menguji 8 posisi tetangga dari posisi (x, y), yaitu posisi yang berada dalam subblok yang sama. Penjelasan dari masing-masing tersebut adalah sbb.:

1. $CheckNeighborCells(r, c, x, y, i)$

   $CheckNeighborCells(r, c, x, y, i)$ adalah fungsi untuk menguji apakah sebuah posisi tetangga atau berada dalam subblok yang sama dengan posisi (x, y) adalah bukan posisi yang tepat untuk meletakkan kotak bernomor i.

2. $isValidPosition(x, y, i)$

   $isValidPosition(x, y, i)$ adalah fungsi untuk menguji apakah posisi (x, y) adalah posisi yang tepat bagi kotak bernomor i berdasarkan aturan model 3.

module *Model 3*

$$
\begin{aligned}
&\mathit{CheckNeighborCells}(r, c, x, y, i) \triangleq \land\ \langle r, c\rangle \neq \langle x, y\rangle \\
&\qquad\qquad\qquad\qquad\qquad\qquad \land\ \mathit{isNeighbor}(r, x) \land \mathit{isNeighbor}(c, y) \\
&\qquad\qquad\qquad\qquad\qquad\qquad \land\ \lor\ \forall k\ :\ \mathit{Board}[k] = \langle r, c, 0\rangle \\
&\qquad\qquad\qquad\qquad\qquad\qquad\quad \lor\ \neg \mathit{isInRow}(r, i) \land \neg \mathit{isInColumn}(c, i) \\
&\mathit{isValidPosition}(x, y, i) \triangleq \land\ \mathit{isEmpty}(x, y) \\
&\qquad\qquad\qquad\qquad\qquad \land\ \neg \mathit{isInRow}(x, i) \land \neg \mathit{isInColumn}(y, i) \land \neg \mathit{isInSubBlock}(x, y, i) \\
&\qquad\qquad\qquad\qquad\qquad \land\ \forall r, c\ :\ \mathit{CheckNeighborCells}(r, c, x, y, i)
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{PutOn}(k) \triangleq \\
&\land\ \neg \mathit{isDone}\ \land k = \mathit{turn} \\
&\land\ \text{IF}\ \exists i\ :\ \mathit{FC}[i][3] = 0 \land\ \mathit{isValidPosition}(\mathit{FC}[i][1], \mathit{FC}[i][2], k) \\
&\quad \text{THEN}\ \exists j\ :\ \land\ \mathit{Board}[j] = \langle \mathit{FC}[i][1], \mathit{FC}[i][2], 0\rangle \\
&\qquad\qquad\ \land\ \mathit{Board}' = [\mathit{Board}\ \text{EXCEPT}\ ![j][3] = k] \\
&\qquad\qquad\ \land\ \mathit{FC}' = [\mathit{FC}\ \text{EXCEPT}\ ![i][3] = k] \\
&\qquad\qquad\ \land\ \mathit{numbers}' = [\mathit{numbers}\ \text{EXCEPT}\ ![\mathit{Board}[j][3]] = \mathit{numbers}[\mathit{Board}[j][3]] - 1] \\
&\qquad\qquad\ \land\ \mathit{success}' = \text{TRUE} \\
&\quad \text{ELSE}\ \text{UNCHANGED}\ \langle \mathit{Board}, \mathit{FC}, \mathit{numbers}\rangle \\
&\land\ \text{IF}\ \mathit{turn} < 9\ \text{THEN}\ \land\ \mathit{turn}' = \mathit{turn} + 1 \\
&\qquad\qquad\qquad\qquad\ \land\ \text{UNCHANGED}\ \langle \mathit{success}, \mathit{isDone}\rangle \\
&\qquad\qquad\qquad \text{ELSE}\ \land\ \text{IF}\ \neg \mathit{success}\ \text{THEN}\ \land\ \mathit{isDone}' = \text{TRUE} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\ \land\ \text{UNCHANGED}\ \langle \mathit{success}, \mathit{turn}\rangle \\
&\qquad\qquad\qquad\qquad\qquad\qquad\ \text{ELSE}\ \land\ \mathit{turn}' = 1 \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\ \land\ \mathit{success}' = \text{FALSE} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\ \land\ \text{UNCHANGED}\ \mathit{isDone}
\end{aligned}
$$

$$
\mathit{TakeBack}(k) \triangleq \text{UNCHANGED}\ v
$$

$$
v \triangleq \langle \mathit{Board}, \mathit{FC}, \mathit{numbers}, \mathit{turn}, \mathit{success}, \mathit{isDone}\rangle
$$

$$
\begin{aligned}
&\mathit{Init} \triangleq \land\ \mathit{turn} = 1 \land \mathit{success} = \text{FALSE} \\
&\qquad\ \land\ \mathit{Board} = ... \\
&\qquad\ \land\ \mathit{FC} = ... \\
&\qquad\ \land\ \mathit{numbers} = ...
\end{aligned}
$$

Gambar 4.8: Spesifikasi Sudoku Model 3.

module *Model 3 - continued*

$Next(k) \triangleq PutOn(k) \lor TakeBack(k)$

$L(k) \triangleq WF_v(PutOn(k)) \land WF_v(TakeBack(k))$

$v \triangleq \langle Board, FC, numbers, turn, isDone, success \rangle$

$Sudoku3 \triangleq \land\ Init$

$\land\ \Box[\exists k \in M\ :\ Next(k)]_v$

$\land\ \forall k \in M\ :\ L(k)$

Gambar 4.9: Spesifikasi Sudoku Model 3 (lanjutan).

Bab 5

Penutup

5.1 Kesimpulan

Kesimpulan yang diambil dari penelitian ini adalah sbb.:

1. Permasalahan Sudoku dapat dimodelkan sebagai *Block−worldproblem* dengan menganalogikan komponen-komponen yang ada pada permasalahan Sudoku dengan komponen-komponen yang ada pada *Block − worldproblem*, yaitu papan sudoku sebagai meja, angka-angka sebagai kotak, dan agen yang bertugas untuk menempatkan angka-angka ke papan Sudoku sebagai robot yang bertugas meletakkan kotak ke meja. Soal Sudoku dianalogikan sebagai konfigurasi awal dari kotak-kotak di meja dan solusi dari permainan Sudoku sebagai konfigurasi akhir.

2. Telah berhasil dikembangkan tiga model untuk permasalahan Sudoku. Permasalahan Sudoku dimodelkan sebagai sistem berparameter (*parameterized system*), dengan model pertama berupa sistem berparameter tak homogen, yaitu masing-masing agen mempunyai tugas/fungsi yang berbeda, dan model ke-2 dan ke-3 berupa sistem berparameter homogen.

3. Telah dihasilkan spesifikasi formal dari model-model yang telah dikembangkan. Spesifikasi menggunakan bahasa spesifikasi TLA (*Temporal Logic of Actions*).

5.2 Rencana pengembangan

Rencana pengembangan atau lanjutan dari penelitian ini adalah

1. mengimplementasikan masing-masing model menjadi prototipe Sudoku *solver*.

2. melakukan eksperimen terhadap Sudoku *solver* yang dihasilkan untuk mengetahui performansi dari ketiga model tersebut.

Referensi

[1] Rhyd Lewis. Metaheuristic can solve Sudoku Puzzles. http://www.inf.utfsm.cl/ mcriff/Tesistas/Games/sudoku.pdf (Akses terakhir 30 Agustus 2012).

[2] Helmut Simonis. Sudoku as a constraints problem. Proc. 4th Int. Works. Modelling and Reformulating Constraint Satisfaction Problems, Brahim Hnich, Patrick Prosser, and Barbara Smith, ed., 2005, pp. 13-27. http://4c.ucc.ie/ brahim/mod-proc.pdf (Akses terakhir 30 Agustus 2012).

[3] Ines Lynce and J. Quaknine. Sudoku as a SAT Problem. Proc. of 9th International Symposium on Artificial Intelligence and Mathematics, January 2006.

[4] Timo Mantere and Janne Koljonen. Solving and Rating Sudoku Puzzles with Genetic Algorithms. http://www.stes.fi/scai2006/proceedings/step2006-86-mantere-solving-and-rating-sudoku-puzzles.pdf (Akses terakhir 30 Agustus 2012).

[5] Cecilia E. Nugraheni. Formal Specification of Parameterized Multi Agent Systems. Proc. of SEAMS GMU 2011.