TESIS-SM 142501

VERIFIKASI FORMAL PETRI NET DENGAN *COUNTER* PADA SISTEM INVENTORI

RUVITA IFFAHTUR PERTIWI
NRP 1214 201 202

DOSEN PEMBIMBING
Dr. Dieky Adzkiya, S.Si., M.Si.
Dr. Subiono, M.S.

PROGRAM MAGISTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

THESIS-SM 142501

FORMAL VERIFICATION PETRI NETS WITH COUNTERS OF INVENTORY SYSTEMS

RUVITA IFFAHTUR PERTIWI
NRP 1214 201 202

SUPERVISORS
Dr. Dieky Adzkiya, S.Si., M.Si.
Dr. Subiono, M.S.

MASTER'S DEGREE
MATHEMATICS DEPARTMENT
FACULTY OF MATHEMATICS AND NATURAL SCIENCES
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2016

VERIFIKASI FORMAL PETRI NET DENGAN *COUNTER* PADA SISTEM INVENTORI

Tesis ini disusun untuk memenuhi salah satu syarat memperoleh gelar Magister Sains (M.Si.)
di
Institut Teknologi Sepuluh Nopember Surabaya

Oleh :
RUVITA IFFAHTUR PERTIWI
NRP. 1214 201 202

Tanggal Ujian : 20 Mei 2016
Periode Wisuda : September 2016

Disetujui oleh :

Dr. Dieky Adzkiya, M.Si.
NIP. 19830517 200812 1 003 **(Pembimbing I)**

Dr. Subiono, M.S.
NIP. 19570411 198403 1 001 **(Pembimbing II)**

Prof. Dr. Erna Apriliani, M.Si.
NIP. 19660414 199102 2 001 **(Penguji)**

Dr. Didik Khusnul Arif, S.Si., M.Si.
NIP. 19730930 199702 1 001 **(Penguji)**

Endah Rokhmati M.P., S.Si., MT., Ph.D.
NIP. 19761213 200212 2 001 **(Penguji)**

Direktur Program Pascasarjana,

Prof. Ir. Djauhar Manfaat, M.Sc., Ph.D.
NIP. 19601202 198701 1 001

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Beberapa masalah yang ditemukan dalam perkembangan sistem *event* diskrit adalah sistem yang berskala sangat besar dimana kardinalitas ruang keadaannya tak berhingga. Kasus ini sering dialami oleh perusahaan atau instansi-instansi yang memiliki sistem dengan data yang sangat banyak, sehingga diperlukan metode yang sistematis untuk mendesain dan menganalisis sistem tersebut. Penanganan sistem dalam permasalahan mengelola data dengan nilai yang banyak, dimana jumlah datanya tidak berhingga dapat diselesaikan, contohnya bagaimana mempertimbangkan model yang efektif dan terstruktur dengan baik untuk merepresentasikan sistem tersebut.

Berdasarkan permasalahan di atas, dalam penelitian ini akan digunakan sebuah kelas Petri net yaitu Petri net dengan *counter* (PNZ) untuk memodelkan suatu sistem. PNZ adalah Petri net yang dilengkapi dengan representasi data dengan variabel-variabelnya adalah bilangan bulat, sehingga transformasi linier dapat diterapkan [5]. Verifikasi formal perlu dilakukan untuk memeriksa apakah sebuah sistem telah memenuhi spesifikasi. Verifikasi formal juga memerlukan adanya spesifikasi formal. Salah satu spesifikasi formal yang sering digunakan adalah *Linear Temporal Logic* (LTL) [2]. LTL merupakan formalisasi *logic* untuk spesifikasi *linier time property* dimana memiliki sintaksis dan semantik yang khusus. Sintaksis merupakan aturan penulisan agar LTL dapat dikonstruksi yang terdiri dari operator Boolean dan operator temporal, sedangkan semantik merupakan arti penulisan dari sintaksis tersebut.

Verifikasi formal dilakukan dengan cara membuat sistem transisi dari model yang akan diverifikasi. Sistem transisi yang berskala sangat besar terkadang memiliki jumlah state tak berhingga, sehingga sistem transisinya menjadi tak berhingga. Hal ini mengakibatkan sistem sulit untuk dianalisis dan diverifikasi. Berdampak juga pada diperlukannya memori yang cukup besar untuk menyimpan atau mengaksesnya, sedangkan kapasitas memori yang dimiliki berhingga. Untuk mengatasi hal tersebut dilakukan abstraksi berhingga pada sistem transisi untuk state yang tak berhingga menjadi berhingga [1]. Verifikasi formal dapat dilakukan secara manual atau otomatis. Verifikasi formal secara otomatis dengan menggu-

nakan *software* lebih efisien karena verifikasi secara manual terkadang sulit untuk dilakukan dan memerlukan waktu yang lama apabila sistem memiliki jumlah state yang tak berhingga. Akan tetapi verifikasi formal secara otomatis hanya dapat dilakukan apabila state nya berhingga, sehingga metode abstraksi berhingga sangat diperlukan.

Berdasarkan uraian yang telah diberikan, pada penelitian ini dibahas kajian untuk melakukan verifikasi formal pada model PNZ suatu sistem inventori dengan menggunakan spesifikasi LTL. Pada penelitian ini akan dikontruksi PNZ dari sistem inventori yang dilakukan oleh Agen berupa pembelian dan penjualan barang. Permasalahan sistem inventori pada penelitian ini memiliki state tak berhingga. Sistem tersebut selanjutnya diubah menjadi sistem transisi, diabstraksi, dan diverifikasi. Kemudian, diimplementasikan menggunakan NuSMV untuk memperoleh hasil verifikasi dari sistem apakah telah memenuhi spesifikasi.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diberikan, maka permasalahan dalam penelitian ini adalah sebagai berikut.

1. Bagaimana memodelkan suatu sistem inventori dengan Petri net dengan *counter* (PNZ)?
2. Bagaimana mengkontruksi abstraksi berhingga Petri net dengan *counter* (PNZ) pada sistem inventori?
3. Apakah Petri net dengan *counter* (PNZ) pada sistem inventori memenuhi beberapa spesifikasi LTL?

1.3 Batasan Masalah

Pada penelitian ini, sistem inventori yang digunakan yaitu dilakukan oleh Agen dengan transaksi pembelian barang dan penjualan barang dengan diberikan batasan masalah sebagai berikut.

1. Barang yang diperjualbelikan ada dua yaitu barang A dan barang B dengan kapasitas satu satuan.
2. Banyak uang yang dimiliki Agen lebih besar dari harga beli barang A atau harga beli barang B.
3. Jumlah marking yang *reachable* adalah 1.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah yang diberikan, maka tujuan dari penelitian ini adalah sebagai berikut.

1. Mengetahui model Petri net dengan *counter* (PNZ) pada sistem inventori.
2. Mendesain metode abstraksi berhingga sistem transisi Petri net dengan *counter* (PNZ) pada sistem inventori yang memiliki jumlah state tak berhingga menjadi berhingga.
3. Memverifikasi formal Petri net dengan *counter* pada sistem inventori menggunakan spesifikasi LTL.

1.5 Manfaat Penelitian

Berdasarkan tujuan penelitian, maka manfaat yang ingin diperoleh adalah sebagai berikut.

1. Diperoleh metode verifikasi Petri net dengan *counter* (PNZ) pada sistem inventori menggunakan spesifikasi LTL.
2. Sebagai salah satu bahan refrensi untuk penelitian selanjutnya khususnya berkaitan pada verifikasi formal dan abstraksi berhingga.
3. Sebagai salah satu kontribusi untuk pengembangan ilmu pengetahuan Matematika dan ilmu Komputer.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

Pada bagian ini dibahas tentang beberapa kajian, teori, serta contoh yang berkaitan dengan penelitian ini untuk memudahkan dalam pembahasan. Uraian kajian dan teori yang dimaksud berkaitan dengan Petri net, verifikasi formal dan abstraksi berhingga. Petri net khusus yang digunakan pada penelitian ini adalah Petri net dengan *counter* dimana diverifikasi formal menggunakan metode abstraksi dan spesifikasi yang digunakan adalah LTL, hal-hal terkait lainnya juga akan diuraikan.

2.1 Penelitian yang Pernah Dilakukan

Beberapa penelitian yang telah dilakukan sebelumnya berkaitan dengan penelitian ini yaitu oleh Franck, Raymond, dan Hanna (2009) pada artikel "*Efficient Reachability Graph Representation of Petri Net With Unbounded Counters*" membahas tentang representasi *reachability* graf pada Petri net dengan *counter* tak berhingga dengan memperkecil state graf pada suatu sistem menjadi berhingga.

Selanjutnya, Adzkiya (2014) pada disertasinya yang berjudul "*Finite Abstraction of Max-Plus-Linear Systems*" membahas tentang metode verifikasi formal sistem max plus linier menggunakan abstraksi berhingga.

Kemudian Marcin, Agnieszka, dan Jerzy (2014) pada artikel "*Methods of Translation of Petri Nets to NuSMV Language*" membahas tentang penerjemahan *reachability* graf dari Petri net ke dalam bahasa NuSMV, dimana NuSMV merupakan alat untuk memudahkan melakukan verifikasi formal dengan spesifikasi CTL dan LTL.

Berdasarkan beberapa penelitian yang telah dilakukan tersebut, pada penelitian ini dibahas mengenai Petri net dengan *counter* yang akan diverifikasi dengan spesifikasi LTL menggunakan abstraksi berhingga dan diimplementasikan pada NuSMV.

2.2 Petri Net

Petri net dikembangkan pertama kali oleh C. A. Petri pada awal 1960-an. Petri net merupakan salah satu alat untuk memodelkan sistem *event* diskrit. Pada Petri net, *event* berkaitan dengan transisi. Beberapa keadaan harus dipenuhi terlebih dahulu agar suatu *event* dapat terjadi. Informasi mengenai *event* dan keadaan ini masing-masing dinyatakan dengan transisi dan *place*. *Place* dapat berfungsi sebagai *input* atau *output* suatu transisi. *Place* sebagai *input* menyatakan keadaan yang

harus dipenuhi agar transisi dapat terjadi. Setelah transisi terjadi maka keadaan akan berubah. *Place* tersebut menyatakan *output* dari transisi [3].

2.2.1 Definisi dan Notasi dalam Petri Net

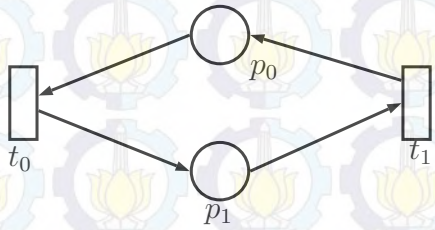
Suatu Petri Net adalah suatu graf bipartisi, yang terdiri dari dua himpunan bagian P dan T, masing-masing menyatakan *place* dan transisi. Petri Net merupakan alat pemodelan yang mudah diaplikasikan untuk banyak sistem. Secara matematis Petri Net dapat dituliskan sebagai 5-*tuple* (P, T, A, w, x) dengan:

- $P = \{p_1, p_2, ..., p_m\}$ adalah himpunan berhingga dari *place*,
- $T = \{t_1, t_2, ..., t_n\}$ adalah himpunan berhingga dari transisi,
- $A \subseteq (P \times T) \cup (T \times P)$ adalah himpunan berhingga dari *arc*,
- $w : A \to \{1, 2, 3, ...\}$ adalah bobot,
- $x : P \to \{1, 2, 3, ...\}$ adalah inisial token, dimana $P \cap T \neq \emptyset$ dan $P \cup T \neq \emptyset$.

Berdasarkan definisi di atas, maka himpunan *place* dan transisi tidak harus berupa himpunan berhingga melainkan bisa berupa himpunan tak hingga terhitung *(countable sets)*. Dalam menjelaskan Petri Net, digunakan notasi $I(t_j)$ untuk merepresentasikan himpunan *place input* untuk transisi t_j, dan notasi $O(t_j)$ untuk merepresentasikan himpunan *place output* untuk transisi t_j. notasi yang sama dapat digunakan untuk merepresentasikan transisi *input* dan *output* untuk *place* p_i, $I(p_i)$ dan $O(p_i)$ [6].

Dalam menggambar Petri net, tipe dibedakan menjadi dua yaitu *place* dan transisi. Dalam tesis ini digunakan lingkaran untuk merepresentasikan sebuah *place* dan persegi panjang untuk merepresentasikan sebuah transisi. *Arcs* yang menghubungkan *place* dan transisi merepresentasikan elemen dari himpunan *arc* A. *Arc* yang menghubungkan *place* p_i ke transisi t_j berarti $p_i \in I(t_j)$. Jika bobot *arc* dari *place* p_i ke transisi t_j adalah k ditulis $w(p_i, t_j) = k$ maka terdapat *arc* sebanyak k dari place p_i ke transisi t_j atau sebuah *arc* dengan bobot k [3].

Contoh 2.1. Diberikan contoh Petri net Sederhana pada Gambar 2.1. Terdapat dua *place* yaitu $P = \{p_0, p_1\}$, dua transisi yaitu $T = \{t_0, t_1\}$, dan 4 *arcs* yaitu $A = \{(t_0, p_1), (p_1, t_1), (t_1, p_0), (p_0, t_0)\}$ dengan masing-masing bobot dari setiap *arcs* adalah $w(t_0, p_1) = w(p_1, t_1) = w(t_1, p_0) = w(p_0, t_0) = 1$. Sedangkan untuk *input* dan *output* dari masing-masing *place* dan transisi dalah sebagai berikut $I(p_1) = t_0,\ O(p_1) = t_1,\ I(p_0) = \{t_1\},\ O(p_0) = \{t_0\},\ I(t_1) = \{p_1\},\ O(t_1) = \{p_0\},\ I(t_0) = \{p_0\},\ O(t_0) = \{p_1\}$.



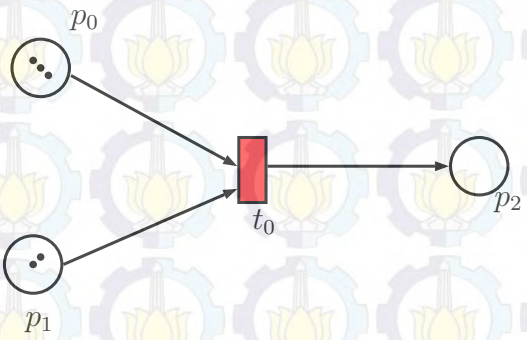
Gambar 2.1: Petri Net Sederhana

2.2.2 Petri Net Bertanda

Transisi pada Petri net merupakan representasi dari kejadian *(event)* pada sistem *event* diskrit *(SED)*, sedangkan *place* merepresentasikan kondisi agar *event* pada sistem dapat terjadi. Dalam hal ini, diperlukan alat sebagai penanda apakah kondisi tersebut dapat terpenuhi atau tidak, dengan cara memberikan token yaitu suatu tanda yang diletakkan pada *place*. Dalam model Petri net, token direpresentasikan dengan dot "•", dan jika jumlah token banyak maka dituliskan dengan angka.

Dalam notasi matematika penanda *(marking)* x pada Petri net bertanda (P, T, A, W, x) adalah fungsi $x : P \rightarrow N = \{0, 1, 2, ...\}$. Penanda dinyatakan dengan vektor kolom yang elemen-elemennya merupakan bilangan bulat tak negatif yang menyatakan banyaknya token dalam suatu *place* yaitu $x = [x(p_1), x(p_2), ..., x(p_n)]^T$. Banyaknya elemen x sama dengan banyaknya *place* dalam Petri net. Elemen ke-i pada vektor ini merupakan banyaknya token pada *place* p_i, dengan demikian $x(p_i) \in \{0, 1, 2, ...\}$. Untuk selanjutnya Petri net bertanda cukup disebut sebagai Petri net saja.

Suatu transisi $t_j \in T$ dalam sebuah Petri net dikatakan kondisinya terpenuhi *(enabled)* jika $x(p_i) \geq w(p_i, t_j)$ untuk setiap $p_i \in I(t_j)$ yang berarti bahwa transisi t_j dalam sebuah Petri net *enabled* jika jumlah token pada *place* p_i sekurang-kurangnya sama dengan bobot pada *arc* yang menghubungkan p_i dengan t_j, untuk semua p_i yang menjadi *input* dari transisi t_j [3].



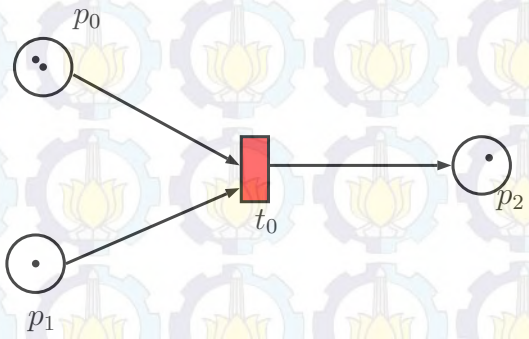
Gambar 2.2: Contoh Transisi *Enable*

Contoh 2.2. Diberikan Petri net pada Gambar 2.2 dengan tiga *place* yaitu p_0, p_1, dan p_2 dan satu transisi yaitu t_0. Transisi berwarna merah berarti *enable*, sedangkan transisi berwarna hitam *disable*. Petri net pada Gambar 2.2 merupakan contoh Petri net dengan transisi yang *enable* dimana ditunjukkan dengan warna merah. Jelas bahwa $I(t_0) = \{p_0, p_1\}, x(p_0) = 3, x(p_1) = 2, w(p_0, t_0) = 1$, dan $w(p_1, t_0) = 1$ seperti yang terlihat Petri net pada Gambar 2.2 transisi t_0 *enable* karena $3 = x(p_0) \geq w(p_0, t_0) = 1$ dan $2 = x(p_1) \geq w(p_1, t_0) = 1$.

2.2.3 Dinamika Petri Net

Mekanisme perubahan keadaan pada Petri Net ditandai dengan perpindahan token-token pada Petri net tersebut sehingga mengubah keadaan Petri net. Secara umum, sebuah transisi pada Petri net yang *enabled*, diistilahkan dengan dapat di-*fire* (*firing* diartikan dengan transisi di-*fire*). Fungsi perubahan keadaan pada Petri net ini merupakan perubahan keadaan Petri net sebelum dan sesudah suatu transisi di-*fire*.

Suatu fungsi perubahan keadaan pada sebuah Petri net (P, T, A, w, x), yaitu $f : \mathbb{N}^n \times T \to \mathbb{N}^n$ terdefinisi untuk transisi $t_j \in T$ jika dan hanya jika $x(p_i) \geq w(p_i, t_j)$ untuk semua $p_i \in I(t_j)$. Jika $f(x, t_j)$ terdefinisi, maka ditulis $x^{'} = f(x, t_j)$ dimana $x^{'}(p_i) = x(p_i) - w(p_i, tj) + w(t_j, p_i), i = 1, 2, ..., n$ yang menyatakan bahwa keadaan tergantung dari fungsi bobot pada setiap *input* dan *output* pada transisi [4].



Gambar 2.3: Contoh Sesudah Transisi t_0 Di-*fire*

Contoh 2.3. Perhatikan jumlah token Petri net di Gambar 2.3. Kondisi sebelum di-*fire* ada pada Petri net di Gambar 2.2 sedangkan Petri net pada Gambar 2.3 merupakan kondisi dimana transisi t_0 di-*fire*. Jelas bahwa Petri net pada Gambar 2.2 dapat di-*fire* dan menjadi Petri net pada Gambar 2.3 karena $3 = x(p_0) \geq w(p_0, t_0) = 1$ dan $2 = x(p_1) \geq w(p_1, t_0) = 1$ menyatakan transisi t_0 *enable* dan dapat di-*fire* dengan perubahan token pada $x(p_0) = 2, x(p_1) = 1$, dan $x(p_2) = 1$.

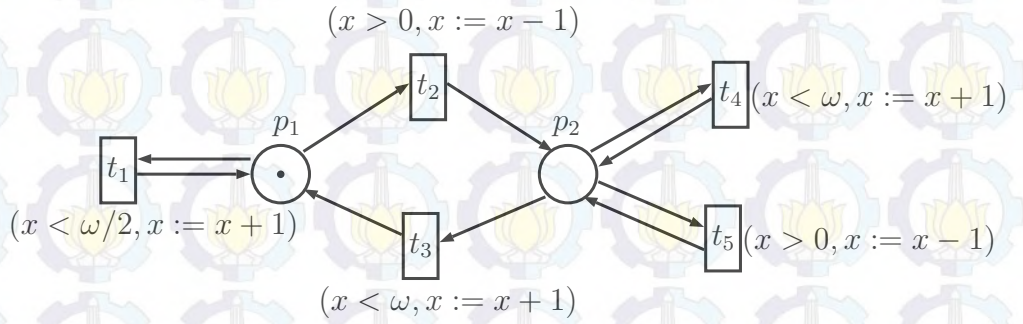
Dengan kata lain, hanya transisi *enabled* yang dapat di-*fire*. Proses yang terjadi pada *firing* transisi menyebabkan semua token di *place input* berkurang sebanyak bobot *arc* yang menghubungkannya dengan transisi yang di-*fire* dan token di *place output* bertambah sebanyak bobot *arc* yang menghubungkannya dengan transisi yang di-*fire*.

2.3 Petri Net dengan *Counter* (PNZ)

Misalkan $\mathbb{Z}$ adalah himpunan dari bilangan bulat dan $\mathbb{D} = \{\bullet\} \uplus \mathbb{Z}$ adalah himpunan dari nilai-nilai data. Diberikan bilangan bulat $n \geq 0$ dan $\mathbb{V} = \{x_0, ..., x_{n-1}\}$ adalah himpunan berhingga dari variabel-variabel sedemikian sehingga $\mathbb{V} \cap \mathbb{D} = \emptyset$. Dipilih asumsi untuk memilih variabel yang dituliskan $X = [x_0, ..., x_{n-1}] \in \mathbb{V}^n$ dimana vektor saling berhubungan satu per satu dengan semua variabel. Sebuah transformasi linier L adalah pasangan (C, U) dimana C adalah kondisi linier dan U adalah *update* linier dengan

- C adalah kondisi linier yang merupakan ekspresi dari $FX \leq Q$, dimana F adalah matriks yang elemen-elemennya bilangan bulat $m \times n$ dan Q adalah vektor di $\mathbb{Z}^m$, untuk $m \geq 0$,
- U adalah *update* linier yang merupakan ekspresi dari $KX + B$, dimana K adalah matriks yang elemen-elemennya bilangan bulat $n \times n$ dan $B \in \mathbb{Z}^n$.

Petri net dengan *counter* (PNZ) adalah Petri net yang dilengkapi dengan representasi data melalui vektor global $V \in \mathbb{Z}^n$ dari nilai-nilai bilangan bulat *(counters)*. Penambahan label dari transisi pada net dengan transformasi linier dapat diperiksa dan *update* data dapat diperoleh pada setiap transisi yang di*fire* [5].



Gambar 2.4: Contoh Petri Net dengan *Counter*

Contoh 2.4. Suatu contoh Petri net dengan *counter* diberikan pada Gambar 2.4, dengan asumsi $X = [x]$, data awal $D_0 = \{[0]\}$ dan setiap transisi dilabeli dengan transformasi linier. Parameter ω adalah bilangan bulat genap tak negatif.

Transformasi linier dari masing-masing transisi adalah

$$L(t_1) = (x < \frac{\omega}{2}, x := x + 1),$$

$$L(t_2) = (x > 0, x := x - 1),$$

$$L(t_3) = (x < \omega, x := x + 1),$$

$$L(t_4) = (x < \omega, x := x + 1),$$

$$L(t_5) = (x > 0, x := x - 1).$$

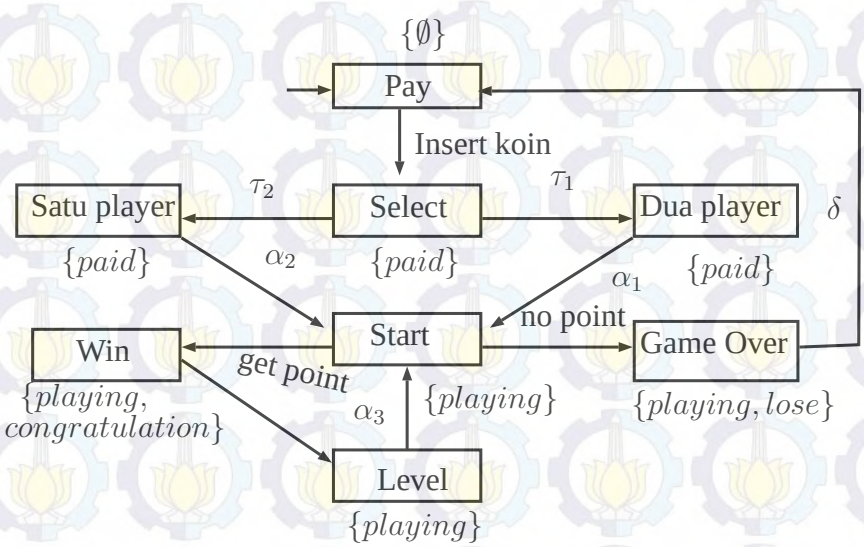
2.4 Verifikasi Formal

Pada penelitian ini akan dilakukan verifikasi formal, dimana verifikasi formal merupakan pengecekan apakah sebuah sistem memenuhi spesifikasi atau tidak. Salah satu teknik yang digunakan adalah model *checking*. Model *checking* dapat dilakukan jika terdapat model dan spesifikasi, dengan prosesnya yaitu memodelkan sistem, membuat spesifikasi formal, selanjutnya mengecek apakah model memenuhi spesifikasi.

Langkah pertama yaitu memodelkan sistem menjadi sistem transisi, dimana sistem pada penelitian ini adalah PNZ. Model standar yang digunakan adalah sistem transisi, sehingga model PNZ nanti akan diubah menjadi sistem transisi. Sistem transisi berguna untuk mendiskripsikan bagaimana dinamika dari sistem secara matematis. Sistem transisi adalah 6- *tuple* $(S, Act, \rightarrow, I, AP, Lb)$ dengan $S = \{s_1, s_2, ...\}$ adalah himpunan dari state-state, *Act* adalah himpunan dari *action*, $\rightarrow S \times Act \times S$ adalah relasi transisi, I adalah himpunan dari Initial state, AP adalah himpunan *atomic proposition*, dan $Lb \rightarrow 2^{AP}$ adalah fungsi labeling, dengan 2^{AP} merupakan himpunan kuasa dari AP [2].

Contoh 2.5. Diberikan contoh dari suatu sistem transisi. Perhatikan sistem transisi pada mesin permainan dingdong berdasarkan Gambar 2.5 diperoleh sistem transisi dengan penjabaran berikut.

- $S = \{pay, select, satu\_player, dua\_player, start, win, level, game\_over\}$,
- $Act = \{insert\_koin, \tau_1, \tau_2, \alpha_1, \alpha_2, \alpha_3, get\_point, no\_point, next\_level, \delta\}$,
- $\rightarrow = \{(pay, insert\_koin, select), ...\}$,
- $I = \{pay\}$,
- $AP = \{paid, playing, congratulation, lose\}$,



Gambar 2.5: Contoh Sistem Transisi pada Mesin Dingdong

- $Lb(pay) = \emptyset$,
  $Lb(select) = \{paid\}$,
  $Lb(satu\_player) = \{paid\}$,
  $Lb(dua\_player) = \{paid\}$,
  $Lb(start) = \{playing\}$,
  $Lb(win) = \{playing, congratulation\}$,
  $Lb(level) = \{playing\}$,
  $Lb(gameover) = \{playing, lose\}$.

Salah satu contoh verifikasi formal yaitu apakah sistem transisi tersebut akhirnya playing adalah benar karena pada state *start* akhirnya akan *playing*.

Sistem transisi pada mesin dingdong diberikan hanya sebagai contoh, tidak berkaitan dengan pembahasan pada penelitian ini. Pada penelitian ini sistem yang akan digunakan adalah sistem inventori.

2.5 *Linear Temporal Logic* (LTL)

Linear Temporal Logic (LTL) adalah salah satu formalisasi *logic* untuk spesifikasi *linear time* (LT properti), dimana LT properti adalah formula *temporal logic* yang mendeskripsikan sebuah barisan tak hingga bernilai benar. Sehingga LTL merupakan sebuah barisan tak hingga dari state. Pada penelitian ini menggunakan spesifikasi LTL karena karakteristik dari LTL yaitu jika sistem transisi abstrak yang dihasilkan dari metode abstraksi berhingga memenuhi spesifikasi, maka sistem transisi kongkrit dari model asli juga pasti memenuhi spesifikasi. Bagaimana menunjukkan LTL dapat digunakan untuk spesifikasi properti sistem didefinisikan

sintaksis dan semantik dari LTL sebagai berikut.

2.5.1 Sintaksis LTL

Sintaksis merupakan aturan penulisan agar formula LTL dapat dikonstruksi. Unsur dasar dari formula LTL adalah *atomic proposition* (label state $a \in AP$). LTL memiliki satu sintaksis yaitu sintaksis path formula.

Definisi 2.1 Sintaksis LTL [2]

Formula LTL atas himpunan dari *atomic proposition* terbentuk mengikuti aturan berikut.

$$\varphi ::= \mathit{true} \mid a \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc \varphi \mid \varphi_1 \cup \varphi_2,\ a \in AP. \tag{2.1}$$

Formula LTL juga dibangun dari operator Boolean seperti $\wedge,\ \vee,\ \leftharpoondown,\ \rightarrow,\ \leftrightarrow$, dan operator temporal seperti yang akan direpresentasikan dalam tabel sebagai berikut.

Tabel 2.1: Sintaksis LTL

Tekstual	Simbolik	Keterangan
X	$\bigcirc$	selanjutnya
G	$\Box$	selalu
F	$\Diamond$	akhirnya
	$\cup$	sampai
	$\wedge$	dan
	$\vee$	atau
	$\rightarrow$	sampai
$X\Phi$	$\bigcirc\Phi$	selanjutnya formula φ
$G\Phi$	$\Box\varphi$	selalu formula φ
$F\varphi$	$\Diamond\varphi$	akhirnya formula φ

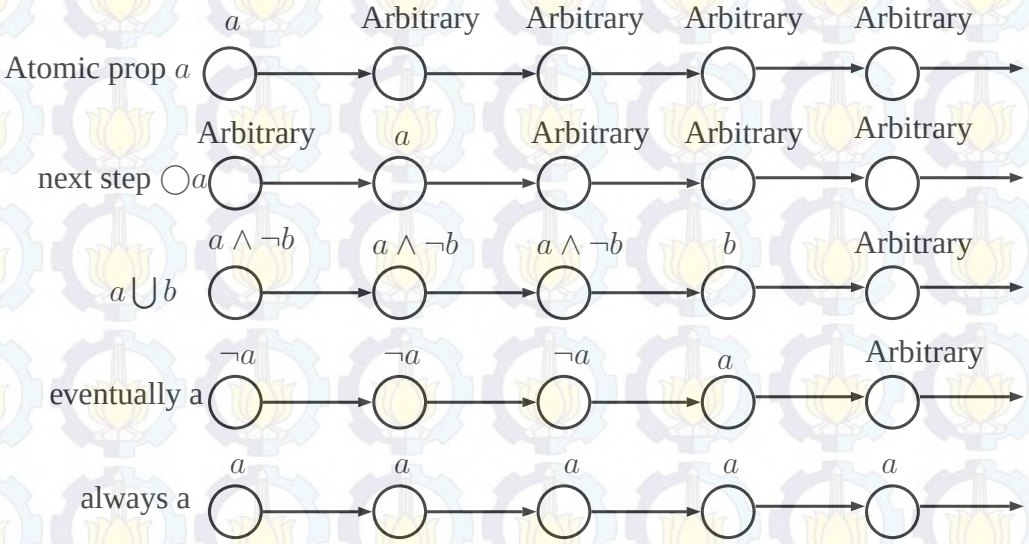
2.5.2 Semantik LTL

Formula LTL dibentuk untuk properti dari path atau *trace* (barisan yang anggotanya adalah himpunan bagian dari AP, menyatakan bahwa apakah path memenuhi formula LTL atau tidak. Semantik dari formula LTL Φ didefinisikan sebagai *word*(φ) yang memuat semua barisan tak hingga atas 2^{AP} yang memenuhi φ. Sehingga semantik adalah interpretasi atas path dan trace pada sebuah sistem transisi.

Definisi 2.2 Semantik LTL untuk Sistem Transisi [2]

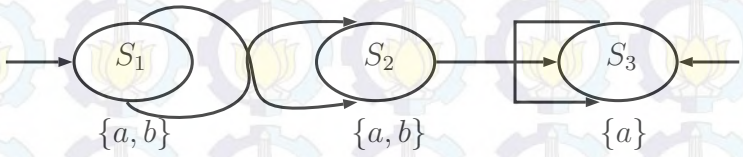
Misalkan $TS = (S, Act, , I, AP, L_b)$ adalah sistem transisi dan φ adalah

formula LTL atas AP. Sistem transisi dapat dikatakan memenuhi spesifikasi apabila $TS \vDash \varphi$. Jadi, $TS \vDash \varphi$ jika dan hanya jika $s_0 \vDash \varphi$ untuk semua inisial state s_0 dari TS.



Gambar 2.6: Semantik dari Operator Temporal

Pada Gambar 2.6 diberikan beberapa semantik dari operator temporal. *atomic prop* a artinya suku atau state yang pertama harus memenuhi a berarti bahwa sebuah path memenuhi LTL formula jika suku pertama dari path tersebut adalah a. Formula $\bigcirc\ a$ menyatakan step selanjutnya atau pada suku ke-dua harus memenuhi a. Formula $a\ \cup\ b$ artinya a harus berlaku dan b tidak berlaku sampai b berlaku, berarti bahwa ada sebuah path yang dimulai dari suku tertentu yang memenuhi a sedemikian sehingga suku-suku sebelumnya tidak memenuhi b. Formula $\diamondsuit\ a$ berarti bahwa ada sebuah path yang dimulai dengan suku tertentu sehingga path disuku-suku sebelumnya tidak memenuhi a. Formula $\square\ a$ artinya a harus selalu terpenuhi [1].



Gambar 2.7: Contoh Sistem Transisi Sederhana

Contoh 2.6. Diberikan sebuah sistem transisi dengan $AP = \{a, b\}$ pada Gambar 2.7, verifikasi formal sebagai berikut.

1. $TS \vDash \square a$ benar, karena semua state dilabeli dengan a.

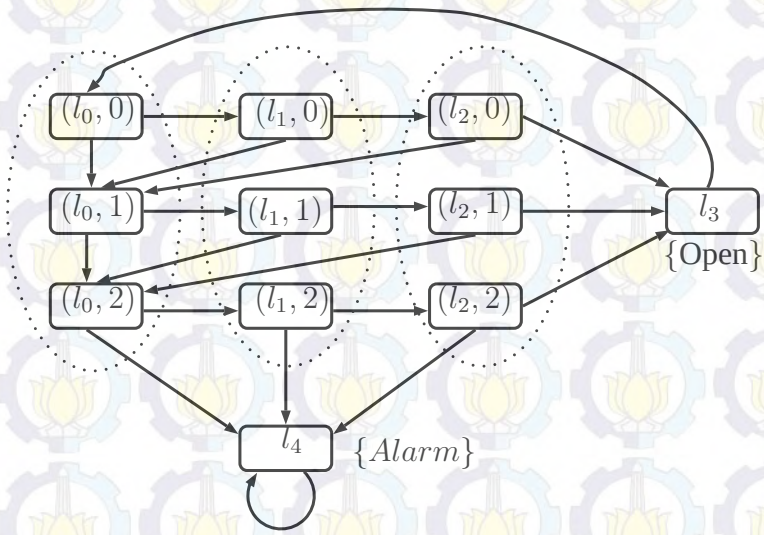
2. $s_1 \vDash \bigcirc(a \wedge b)$ benar, karena $s_1 \vDash (a \wedge b)$.

2.6 Abstraksi Berhingga

Abstraksi berhingga merupakan konsep yang digunakan untuk menganalisa atau memverifikasi sistem transisi dengan state yang besar atau tak berhingga. Abstraksi berhingga membuat state tak berhingga menjadi berhingga atau membuat sistem transisi dengan jumlah state yang besar menjadi sistem transisi dengan jumlah state yang lebih kecil. Abstraksi didefinisikan sebagai sebuah himpunan dari state abstrak $\hat{S}$, fungsi abstrak f yang menghubungkan setiap state kongkrit $S \rightarrow \hat{S}$ pada sistem transisi ke state abstrak yang direpresentasikan dengan $f(s)$, dan himpunan AP dari *atomic propositions* sebagai label dari state kongkrit dan state abstrak [1].

Abstraksi berhingga diperoleh dengan menggabungkan himpunan dari beberapa state kongkrit ke suatu state abstrak. Fungsi abstraksi memetakan state kongkrit ke suatu state abstrak, sedemikian sehingga state abstrak menghubungkan state kongkrit yang memiliki label yang sama. Sifat-sifat dari abstraksi salah satunya adalah jika abstraksi tidak terpenuhi, maka tidak dapat disimpulkan sistem transisi kongkrit nya juga tidak terpenuhi. Tetapi, jika sistem transisi abstrak memenuhi spesifikasi, maka sistem transisi kongkritnya juga memenuhi spesifikasi

Abstraksi berhingga dilakukan dengan mempartisi state *space* menjadi beberapa kelas ekuivalen, dimana setiap kelas memiliki label (AP) yang sama, kemudian menentukan transisi pada sistem transisi abstrak. Adanya transisi dari state abstrak $f(s)$ ke state $f(s')$ jika ada transisi dari s ke s' dengan fungsi abstrak yaitu $f : S \rightarrow \hat{S}$ [2].



Gambar 2.8: Sistem Transisi Pintu Otomatis

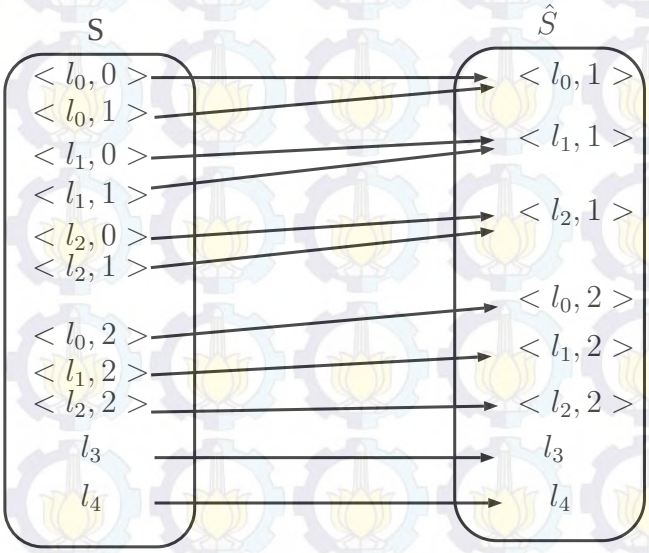
Contoh 2.7. Diberikan contoh abstraksi berhingga dari jumlah state yang berhingga yaitu sistem transisi pintu otomatis diperlihatkan pada Gambar 2.8.

Diketahui pintu otomatis dengan AP $= \{$*alarm*,*open*$\}$. Kemudian diberikan fungsi f untuk abstraksi

$$f(< \ell, error = k >) = \begin{cases} < \ell, error \leq 1 > jika & k \in \{0, 1\}, \\ < \ell, error \leq 2 > jika & k \in 2. \end{cases}$$

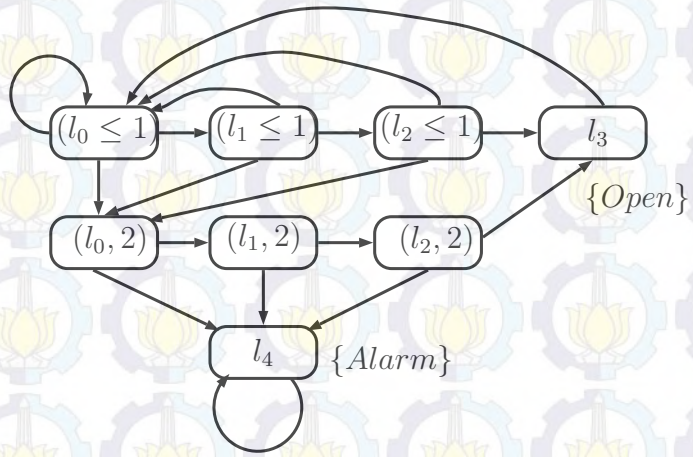
Pintu akan terbuka dengan 3 digit kode ℓ_1, ℓ_2, ℓ_3 dengan $\ell_{1,2,3} \in \{0, ..., 9\}$ dan ℓ_i untuk $i = 0, 1, 2$. Digit i pertama menyatakan kode yang dimasukkan benar, komponen kedua menyatakan berapa kali melakukan kesalahan. Salah satu contohnya adalah state $(\ell_0, 1)$ dimana kode yang dimasukkan adalah salah yang berarti bahwa terjadi satu kali kesalahan. Apabila keadaan itu terjadi, maka tersisa dua kesempatan untuk melakukan kesalahan sebelum alarm berbunyi.

Setelah mengetahui fungsi abstraksi, state pada sistem transisi pintu otomatis dipartisi sesuai dengan f, masing-masing state dipetakan ke state abstrak.



Gambar 2.9: Contoh Fungsi Abstraksi

Selanjutnya dilakukan abstraksi berdasarkan fungsi abstrak. Partisi pada sistem transisi diperoleh pada pada Gambar 2.9, diketahui state pada sistem transisi pada Gambar 2.8 jumlahnya berkurang, sehingga sistem transisi pintu otomatis menjadi



Gambar 2.10: Sistem Transisi Abstrak Pintu Otomatis

2.7 NuSMV

Verifikasi formal dapat dilakukan secara otomatis dengan menggunakan *software* model *checker*. NuSMV adalah salah satu model *checker* untuk *temporal logic*. Jika diberikan sebuah model dengan state yang berhingga dan satu atau lebih formula, NuSMV dapat digunakan untuk memeriksa secara otomatis apakah model memenuhi spesifikasi tersebut atau tidak. Formula dapat diekpresikan dengan menggunakan spesifikasi LTL. Pada NuSMV apabila sitem tidak memenuhi spesifikasi diberikan *counterexample* agar kita mengetahui state mana yang tidak memenuhi. Hasil yang diberikan NuSMV adalah *true* atau *false* yang menyatakan bahwa apakah sitem memenuhi spesifikasi.

Pada NuSMV, sistem yang diverifikasi adalah model sistem transisi dengan state berhingga. Sehingga sistem dapat dideskripsikan sebagai *tuple* $TS = (S, I, \rightarrow, L_b)$, dimana S adalah himpunan dari state, $I \subseteq S$ adalah himpunan dari initial state,$\rightarrow \subseteq S \times S$ adalah relasi transisi, dan Lb adalah fungsi *labeling* [7].

NuSMV memiliki model bahasa sendiri yang digunakan untuk mendefinisikan state berhingga pada sistem transisi. Berikut akan diberikan contoh sistem transisi dengan bahasa NuSMV.

Contoh 2.8. Pada Gambar 2.11 model NuSMV terdiri dari beberapa bagian yaitu VAR dan ASSIGN. Bagian VAR mengandung definisi dari variabel, termasuk himpunan dari state dan variabel *atomic propositions*. Bagian ASSIGN terbagi lagi menjadi tiga bagian, yang pertama inisialisasi variabel state. Selanjutnya, bagian kedua adalah respon transisi antara state-state dan yang ketiga adalah nilai *atomic propositions* untuk state spesifik. Bagian yang terakhir yaitu spesifikasi.

MODULE main
VAR
state : $\{s1, s2, s3, s4\}$; — set of state

ASSIGN
init(state):=s1; — initial state
next(state) := case
state = s4 : $\{s2, s3\}$;
state = s3 : $\{s1, s2, s4\}$;
state = s2 : $\{s1, s2, s4\}$; — transition relation
state = s1 : $\{s2, s3\}$
TRUE : $\{s1, s2, s3, s4\}$;
esac;

DEFINE
g1 := $state = s1 \mid state = s2$; — labelling function
g2 := $state = s1 \mid state = s3$;

LTLSPEC X (G g1);
LTLSPEC F (G g2); — specification

Gambar 2.11: Contoh Sistem Transisi dengan Bahasa NuSMV

2.8 Inventori (Persediaan)

Inventori atau persediaan adalah bahan atau barang yang disimpan yang akan digunakan untuk memenuhi tujuan tertentu, misalnya untuk produksi atau untuk dijual kembali. Setiap perusahaan perlu mengadakan persediaan untuk menjamin kelangsungan hidup usahanya. Oleh karena itu, setiap perusahaan haruslah mempertahankan suatu jumlah persediaan optimum yang dapat menjamin kebutuhan bagi kelancaran kegiatan perusahaan dalam jumlah dan mutu yang tepat dengan biaya rendah. Mengatur tersedianya suatu tingkat persediaan yang optimum, maka diperlukan suatu sistem persediaan dengan tujuan menjaga jangan sampai kehabisan persediaan yang mengakibatkan terhentinya kegiatan usaha.

Secara umum inventori berfungsi untuk mengolah persediaan barang dagangan yang selalu mengalami perubahan jumlah dan nilai melalui transaksi-transaksi pembelian dan penjualan. Bila melakukan kesalahan dalam menetapkan besarnya persediaan, maka akan berdampak pada tidak terpenuhinya permintaan konsumen atau bahkan berlebihan persediaan sehingga tidak semua terjual sehingga ada biaya tambahan untuk penyimpan. Pada Penelitian ini digunakan suatu permasalahan inventori dengan transaksi pembelian barang, penyimpanan barang dalam gudang, dan penjualan barang.

BAB III
METODE PENELITIAN

Berikut diberikan beberapa tahapan penelitian yang akan dilakukan dalam penelitian yang diusulkan.

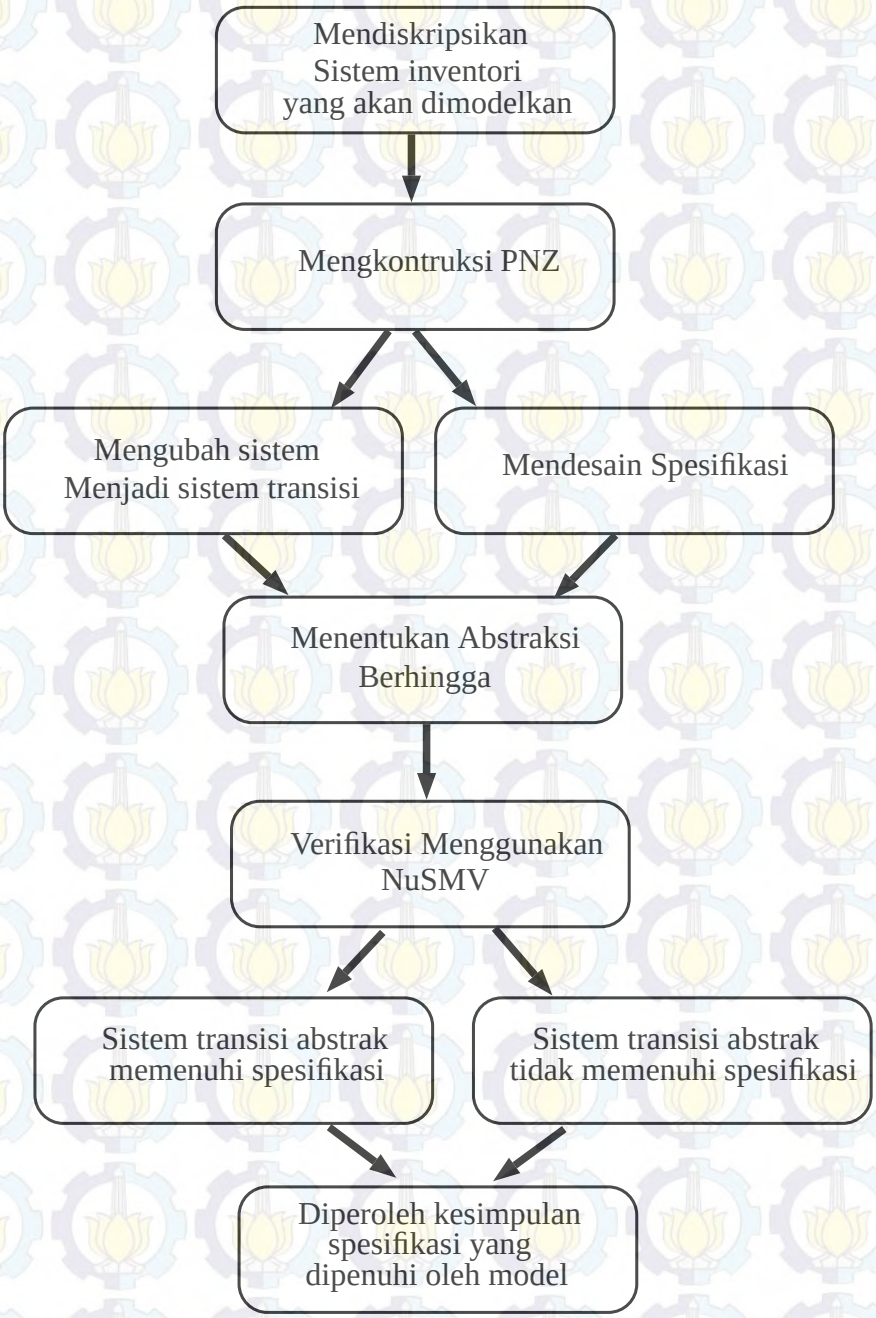
3.1 Tahapan Penelitian

Penelitian ini dapat dibagi menjadi beberapa tahap sebagai berikut.

1. Studi literatur dengan mengkaji teori mengenai Petri net dengan *counter* serta beberapa hal yang terkait untuk pembahasan. Pengkajian dilakukan dengan mengumpulkan beberapa informasi pada jurnal-jurnal yang terkait.
2. Mengkontruksi sebuah Petri net dengan *counter* sistem inventori.
3. Mengubah model sistem menjadi sistem transisi.
4. Mendesain spesifikasi formal LTL.
5. Melakukan abstraksi berhingga untuk memperoleh state yang berhingga.
6. Memverifikasi sistem dengan implementasi pada NuSMV.
7. Menganalisis hasil verifikasi. Pada tahap ini dapat diketahui apakah model memenuhi spesifikasi atau tidak.
8. Publikasi karena usulan penelitian yang telah dilakukan selanjutnya diseminarkan pada suatu seminar internasional.
9. Penulisan Tesis dilakukan dengan penulisan laporan hasil penelitian yang dilakukan mulai tahap studi literatur sampai hasil implementasi.

3.2 Diagram Alir Penelitian

Berikut merupakan gambar diagram alir penelitian ini.



Gambar 3.1: Diagram Alir Penelitian

BAB IV
HASIL DAN PEMBAHASAN

Pada bagian ini dibahas tentang memodelkan Petri net dengan *counter* (PNZ) sistem inventori dan memverifikasi formal Petri net dengan *counter* (PNZ) sistem inventori menggunakan metode abstraksi berhingga. Diberikan deskripsi permasalahan sistem inventori yang digunakan, langkah-langkah memverifikasi, abstraksi berhingga, dan desain spesifikasi menggunakan spesifikasi LTL. Selain itu diberikan juga hasil implementasi pada NuSMV.

4.1 Pendahuluan

Model sistem yang digunakan untuk verifikasi formal adalah sistem transisi. Sistem transisi digunakan untuk mendeskripsikan bagaimana dinamika dari sistem. Apabila sistem yang akan diverifikasi belum berupa sistem transisi, maka sistem tersebut dibuat sistem transisinya. Pada penelitian ini dikaji tentang bagaimana memverifikasi formal PNZ pada sistem inventori.

Sistem yang akan digunakan pada penelitian ini adalah suatu sistem inventori (persediaan) yang dilakukan oleh Agen berupa pembelian dan penjualan barang. Permasalahan sistem persediaan pada penelitian ini yaitu untuk mewakili sistem berskala besar apabila terdapat banyaknya transaksi pembelian barang atau penjualan barang, sehingga memiliki jumlah state yang tak berhingga. Sistem inventori tersebut dibuat dalam bentuk PNZ.

Setelah mengkontruksi PNZ, dibuat suatu sistem transisi. Sistem transisi tersebut memiliki state tak berhingga untuk menggambarkan suatu sistem yang berskala besar, sehingga diperlukan abstraksi berhingga untuk membuat state menjadi berhingga. Apabila sistem transisi abstrak telah memiliki state yang berhingga, dilakukan verifikasi dengan spesifikasi LTL. Verifikasi PNZ pada sistem inventori dapat diimplementasikan menggunakan *software* NuSMV untuk mengetahui apakah sistem yang dimiliki sesuai dengan spesifikasi. Pada penelitian ini metode-metode abstraksi dan verifikasi akan diterapkan pada PNZ dengan permasalahan sistem inventori seperti yang telah dipaparkan, sehingga contoh-contoh yang diberikan pada penelitian ini memiliki keterkaitan. Sebelumnya, perlu diketahui definisi state dan state *space* dari PNZ sebagai berikut.

Definisi 4.1 Petri Net dengan *Counter* (PNZ) [5]

PNZ adalah 4-*tuple* (P, T, ℓ, L) dengan P adalah himpunan berhingga dari *place*, T adalah himpunan berhingga dari transisi, ℓ adalah label fungsi pada *place*, dan L adalah pemetaan yang menghubungkan setiap transisi t di T sebagai sebuah transformasi linier $L = (C, U)$.

Diberikan juga definisi state dan state *space* dari PNZ sebagai berikut.

Definisi 4.2 State dan State *Space* PNZ

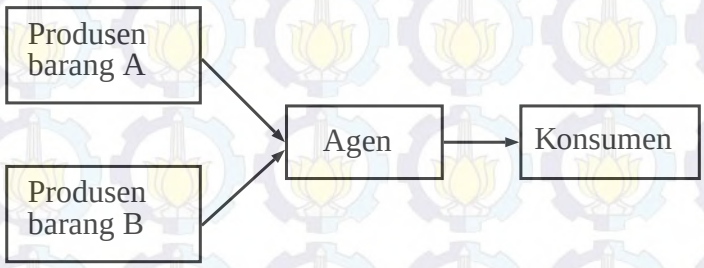
Sebuah state dari PNZ dinotasikan (M, D) dimana M adalah *marking* yang menyatakan jumlah token di *place* dan D adalah data yang menyatakan nilai dari setiap variabel.

- $M \in \mathbb{Z}_{\geq 0}^{|P|}$ dengan $|P|$ adalah banyaknya *place* di PNZ,
- $D \in \mathbb{Z}^n$ dengan n adalah banyaknya variabel,

sedemikian sehingga state $(M, D) \in \mathbb{Z}_{\geq 0}^{|P|} \times \mathbb{Z}^n$, dimana $\mathbb{Z}_{\geq 0}^{|P|} \times \mathbb{Z}^n$ merupakan state *space* pada PNZ.

4.2 PNZ pada Sistem Inventori

Dikontruksi PNZ dari sistem inventori yang dilakukan oleh seorang Agen. PNZ ini akan merepresentasikan proses membeli dan menjual barang yang dilakukan oleh seorang Agen yang memiliki dua produsen barang A dan barang B. Setelah pembelian barang, Agen menjual barang tersebut kembali kepada konsumen dengan harga yang lebih besar dari harga belinya.



Gambar 4.1: Diagram Kompartemen Inventori oleh Agen

Pada sistem ini hanya terdapat dua barang yaitu barang A dan barang B dengan kapasitas satu satuan. Tidak terdapat biaya pemesanan dan penyimpanan. Variabel-variabel yang digunakan sebagai berikut.

- x_1 adalah banyaknya uang yang dimiliki Agen,
- x_2 adalah banyaknya barang A,
- x_3 adalah banyaknya barang B.

Beberapa konstanta digunakan yaitu

- a adalah harga beli barang A,
- b adalah harga jual barang A,
- c adalah harga beli barang B,
- d adalah harga jual barang B,
- ω adalah kapasitas gudang.

Berdasarkan permasalahan sistem inventori di atas, kondisi linier dan *update* linier diberikan sebagai berikut.

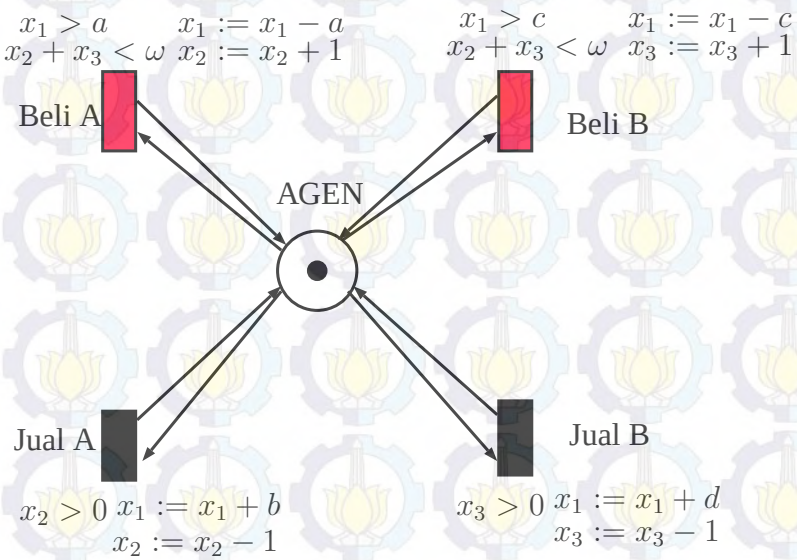
1. Transisi Beli A dapat di*fire* jika memenuhi kondisi $x_1 > a$ dan $x_2 + x_3 < \omega$ yang berarti bahwa uang yang dimiliki Agen harus lebih besar dari harga beli barang A dan jumlah barang yang dimiliki kurang dari kapasitas gudang. Data yang ter*update* adalah $x_1 := x_1 - a$ yang berarti jumlah uang yang dimiliki berkurang sebesar harga beli barang A, sedangkan $x_2 := x_2 + 1$ berarti banyaknya barang A yang dimiliki bertambah.

2. Transisi Jual A dapat di*fire* jika memenuhi kondisi $x_2 > 0$ yang artinya adalah harus terdapat barang A dalam gudang. Data yang ter*update* adalah $x_1 := x_1 + b$ yang berarti jumlah uang yang dimiliki bertambah sebesar harga jual A, sedangkan $x_2 := x_2 - 1$ berarti banyaknya barang A yang dimiliki berkurang.

3. Transisi Beli B dapat di*fire* jika memenuhi kondisi $x_1 > c$ dan $x_2 + x_3 < \omega$ yang berarti bahwa uang yang dimiliki Agen harus lebih besar dari harga beli barang B dan jumlah barang yang dimiliki kurang dari kapasitas gudang. Data yang ter*update* adalah $x_1 := x_1 - c$ yang berarti jumlah uang yang dimiliki berkurang sebesar harga beli barang B, sedangkan $x_3 := x_3 + 1$ berarti banyaknya barang B yang dimiliki bertambah.

4. Transisi Jual B dapat di*fire* jika memenuhi kondisi $x_3 > 0$ yang artinya adalah harus terdapat barang B dalam gudang. Data yang ter*update* adalah $x_1 := x_1 + d$ yang berarti jumlah uang yang dimiliki bertambah sebesar harga jual B, sedangkan $x_3 := x_3 - 1$ berarti banyaknya barang B yang dimiliki berkurang.

Sehingga PNZ dari sistem inventori sebagai berikut



Gambar 4.2: PNZ Sistem Inventori

Contoh 4.1. PNZ dari sistem inventori pada Gambar 4.2 terdiri dari satu *place* dan empat transisi yaitu transisi Beli A, Jual A, Beli B, dan Jual B. Terdapat 1 token pada *place* yang berarti bahwa jumlah *marking* yang *reachable* adalah satu dan token menunjukkan state $(1, [x_1, x_2, x_3])$. Setiap transisi dilabeli dengan kondisi linier dan *update* linier yang nilainya bilangan bulat tak negatif.

Transisi Beli A dan Beli B berwarna merah pada PNZ dari sistem inventori dalam Gambar 4.2. Hal ini menyatakan bahwa kemungkinan transisi pertama yang *enable* dan dapat di*fire* adalah transisi Beli A atau Beli B. Sedangkan transisi Jual A dan Jual B berwarna hitam berarti bahwa transisi tersebut tidak *enable*, hal ini karena pada transaksi pertama kali kondisi linier pada transisi Jual A atau Jual B tidak terpenuhi sehingga Agen tidak dapat menjual barang A atau B.

4.3 Sistem Transisi

Sistem transisi bermula dari *Initial* state $(s_0 \in I)$ dan dinamikanya diberikan oleh relasi transisi. Jika sebuah state memiliki lebih dari satu tujuan, transisi selanjutnya dipilih berdasarkan kondisi yang telah dipenuhi. Pada saat mendesain

sistem transisi dari suatu sistem, perlu dilakukan penyesuaian pada sistem agar dapat diubah menjadi sistem transisi. Definisi sistem transisi telah diberikan pada bagian sebelumnya, berikut diberikan definisi transisi yang disesuaikan dengan PNZ.

Definisi 4.3. Sistem Transisi yang Dibangun oleh PNZ

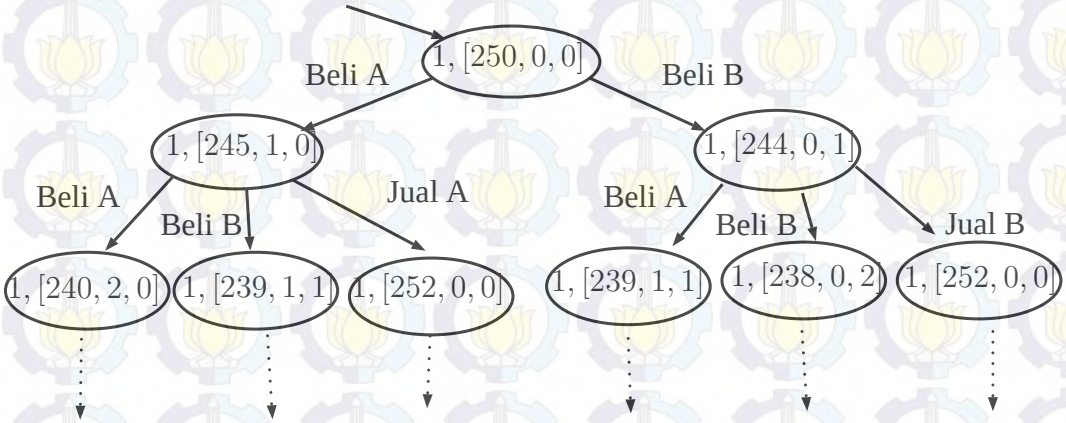
Sistem transisi yang dibangun oleh PNZ adalah $(S, \rightarrow, I, AP, L_b)$ dimana

- himpunan dari state S adalah $\{(M, D)\} \in \mathbb{Z}_{\geq 0}^{|P|} \times \mathbb{Z}^n$,
- terdapat relasi transisi $(M, D) \longrightarrow (M', D')$ jika ada transisi t_j yang memenuhi kondisi
  1. $FD \leq Q$ yang menyatakan kondisi linier dari PNZ untuk t_j.
  2. $M_i \geq w(p_i, t_j)$ untuk setiap $p_i \in I(t_j)$.
  3. $M' = M + Ae_j$ dengan e_j merupakan matriks kolom ke-j dari matriks identitas berorder $|P|$ dan matriks *incidence* $A = A_f - A_b$,
  4. $D' = KD + B$ yang menyatakan *update* linier dari PNZ untuk transisi t_j.

  Artinya, terdapat transisi pada PNZ yang di*fire* sehingga mengakibatkan marking dan data berubah dari $(M, D) ke (M', D')$.
- himpunan dari *initial* state *I* adalah $\{(M_0, D_0)\}$.
- himpunan AP diasumsikan berhingga,
- dan diasumsikan $L_b^{-1}(a) = \{(M, D) | F_i D \leq Q_i\}$.

Contoh 4.2. Diberikan sistem transisi yang dibangun oleh PNZ sistem inventori yang ada pada Gambar 4.2. State dari PNZ berbentuk (M, D) dengan M adalah *marking* dari dari *place* p dan $D = [x_1, x_2, x_3]$.

Misalkan diberikan nilai awal pada masing-masing variabel yaitu $x_1 = 250, x_2 = 0, x_3 = 0$ dan kapasitas penyimpanan $\omega = 40$, harga beli A adalah $a = 5$, harga jual A adalah $b = 7$, harga beli B adalah $c = 6$, dan harga jual B adalah $d = 8$. Sistem transisi pada Gambar 4.3 memiliki *Initial* state (M_0, D_0) dengan $M_0 = 1$ dan $D_0 = [250, 0, 0]$ yang menjelaskan kondisi awal tentang seorang Agen memiliki uang sejumlah 250. State yang dapat terjadi yaitu Beli Produk A atau Beli Produk B, sehingga transisi dari state $(1, [250, 0, 0])$ dapat menuju ke state $(1, [245, 1, 0])$ atau state $(1, [244, 0, 1])$.



Gambar 4.3: Contoh Sistem Transisi yang Dibangun dari PNZ

Misalkan dari *initial* state selanjutnya adalah Beli A maka statenya menjadi $(1, [245, 1, 0])$, selanjutnya terdapat transisi yang *enable* yaitu Beli A, Beli B, atau Jual A. Akibatnya, jika Beli A atau Beli B uang yang dimiliki berkurang tetapi jumlah barang A atau barang B bertambah. Berbeda apabila setelah Beli A kemudian Jual A, uang bertambah tetapi tidak terdapat barang pada gudang karena habis terjual. Begitu seterusnya seperti yang ada pada Gambar 4.3

Sistem transisi pada Gambar 4.3 merupakan sistem transisi dengan state tak berhingga. Misalkan diberikan contoh transisi pada PNZ yang di*fire* hanya Beli A dan Jual A secara terus menerus yang berarti bahwa Agen hanya membeli dan menjual barang A untuk mendapatkan keuntungan, sehingga uang yang dimiliki (x_1) jumlahnya bertambah. State tidak kembali ke state yang awal karena nilai data pada state berubah-ubah dan bisa menjadi tak berhingga.

4.4 Abstraksi Berhingga

Abstraksi berhingga merupakan cara yang digunakan untuk menganalisa atau memverifikasi sistem dengan state yang besar atau tak berhingga. Misalkan sistem transisi (TS) dengan jumlah state yang besar menjadi TS dengan jumlah state yang lebih kecil, sedangkan untuk TS dengan jumlah state yang tak berhingga menjadi TS dengan jumlah state yang berhingga.

State kongkrit adalah state dari sistem transisi kongkrit. Sistem transisi kongkrit adalah sistem transisi asli atau langsung dari model yang akan diverifikasi tanpa ada tambahan metode apapun. Sistem transisi kongkrit mencerminkan dinamika dari model. Pada penelitian ini sistem transisi kongkrit adalah sistem transisi asli dari PNZ pada sistem inventori.

Abstraksi berhingga didefinisikan sebagai sebuah himpunan state-state yang

abstrak $\hat{S}$ dengan fungsi abstraksi $f(s)$ yang menghubungkan setiap state yang kongkrit dengan state yang abstrak, dan himpunan AP sebagai label dari state yang kongkrit dan state yang abstrak, dimana state kongkrit merupakan state dari PNZ, sedangkan state abstrak dari state PNZ setelah diabstraksi.
Tata cara abstraksi berhingga pada PNZ adalah sebagai berikut.

1. Mempartisi state *space* dari PNZ menjadi beberapa kelas ekivalen, dimana setiap kelas ekivalen memiliki label (AP) yang sama.

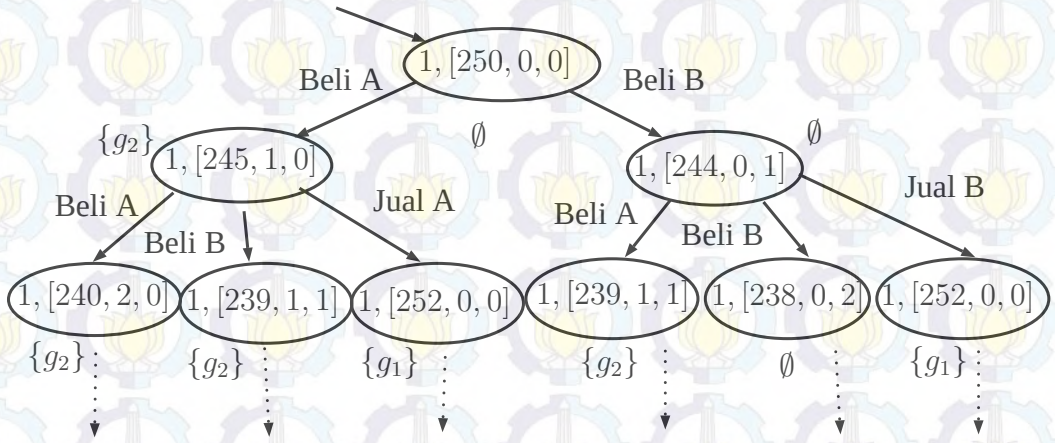
2. Menentukan transisi dari setiap state kongkrit ke state abstrak.

Himpunan state yang abstrak berisi maksimal dengan jumlah state adalah $2^{|AP|}$, dimana $|AP|$ menyatakan jumlah anggota dari himpunan AP.

4.4.1 Menentukan State pada Sistem Transisi Abstrak

Membangun partisi dari S dengan fungsi abstraksi f yang memetakan setiap state yang memiliki label sama ke state abstrak tertentu. Langkah-langkahnya yaitu:

- menentukan AP dari PNZ,
- melabeli setiap state PNZ dengan AP yang sesuai,
- mempartisi state sesuai label yang sama,
- diperoleh state abstrak.



Gambar 4.4: Pelabelan Sistem Transisi PNZ

Contoh 4.3. Diberikan AP dari Contoh 4.2 yaitu $AP = \{g_1, g_2\}$ dengan himpunan state yang memenuhi g_1 adalah $\{x|x_1 > 250\}$ sedangkan himpunan state yang memenuhi g_2 adalah $\{x|x_2 > 0\}$, dimana g_1 berarti mendapat keuntungan dan

g_2 berarti tersedianya barang A dalam gudang. Misalkan diambil state-state dari Contoh 4.2 yaitu
$(1,[250,0,0]) = s_1$, $(1,[245,1,0]) = s_2$,
$(1,[244,0,1]) = s_3$, $(1,[240,2,0]) = s_4$,
$(1,[239,1,1]) = s_5$, $(1,[252,0,0]) = s_6$,
$(1,[239,1,1]) = s_7$, $(1,[238,0,2]) = s_8$,
$(1,[252,0,0]) = s_9$, dan seterusnya,
sehingga label (L_b) dari state pada sistem transisi sebagai berikut:
$Lb(s_1) = \emptyset$, $Lb(s_2) = \{g_2\}$,
$Lb(s_3) = \emptyset$, $Lb(s_4) = \{g_2\}$,
$Lb(s_5) = \{g_2\}$, $Lb(s_6) = \{g_1\}$,
$Lb(s_7) = \{g_2\}$, $Lb(s_8) = \emptyset$,
$Lb(s_9) = \{g_2\}$, dan seterusnya.

Berdasarkan AP yang diketahui terdapat empat partisi yaitu $\bar{S}_1, \bar{S}_2, \bar{S}_3$, dan $\bar{S}_4$ dsebagai berikut.

- $g_1 \wedge g_2 = \bar{S}_1 = \{(1,[x_1,x_2,x_3])|x_1 > 250, x_2 > 0\}$
- $g_1 \wedge \leftharpoondown g_2 = \bar{S}_2 = \{(1,[x_1,x_2,x_3])|x_1 > 250, x_2 \leq 0\}$
- $\leftharpoondown g_1 \wedge g_2 = \bar{S}_3 = \{(1,[x_1,x_2,x_3])|x_1 \leq 250, x_2 > 0\}$
- $\leftharpoondown g_1 \wedge \leftharpoondown g_2 = \bar{S}_4 = \{(1,[x_1,x_2,x_3])|x_1 \leq 250, x_2 \leq 0\}$

Selanjutnya didefinisikan fungsi abstrak $f : S \to \hat{S}$ sedemikian sehingga

$$f(s) = \begin{cases} \hat{S}_1 & \text{jika } s \in \bar{S}_1, \\ \hat{S}_2 & \text{jika } s \in \bar{S}_2, \\ \hat{S}_3 & \text{jika } s \in \bar{S}_3, \\ \hat{S}_4 & \text{jika } s \in \bar{S}_4. \end{cases} \tag{4.1}$$

Berdasarkan partisi diperoleh empat state abstrak yaitu $\hat{S}_1, \hat{S}_2, \hat{S}_3$, dan $\hat{S}_4$ sehingga $f : \bar{S} \to \hat{S}$. State-state dari sistem transisi awal akan dipartisi kemudian dipetakan ke state abstrak.

- s_1 memenuhi $\bar{S}_4$ sehingga $s_1 \in \bar{S}_4$, maka $f(s_1) \in \hat{S}_4$,
- s_2 memenuhi $\bar{S}_3$ sehingga $s_2 \in \bar{S}_3$, maka $f(s_2) \in \hat{S}_3$,
- s_3 memenuhi $\bar{S}_4$ sehingga $s_3 \in \bar{S}_4$, maka $f(s_3) \in \hat{S}_4$,

- s_4 memenuhi $\bar{S}_3$ sehingga $s_4 \in \bar{S}_3$, maka $f(s_4) \in \hat{S}_3$,
- s_5 memenuhi $\bar{S}_3$ sehingga $s_5 \in \bar{S}_3$, maka $f(s_5) \in \hat{S}_3$,
- s_6 memenuhi $\bar{S}_2$ sehingga $s_6 \in \bar{S}_2$, maka $f(s_6) \in \hat{S}_2$,
- s_7 memenuhi $\bar{S}_3$ sehingga $s_7 \in \bar{S}_3$, maka $f(s_7) \in \hat{S}_3$,
- s_8 memenuhi $\bar{S}_4$ sehingga $s_8 \in \bar{S}_4$, maka $f(s_8) \in \hat{S}_4$,
- s_9 memenuhi $\bar{S}_2$ sehingga $s_9 \in \bar{S}_2$, maka $f(s_9) \in \hat{S}_2$,
- dan seterusnya.

Semua state dapat dipetakan ke state abstrak, tetapi pada contoh ini hanya ditunjukkan sebagian dari state. Hal ini menunjukkan bahwa setiap partisi bukan himpunan kosong ϕ karena terdapat state-state yang memenuhi kondisi partisi. Sehingga akan diperoleh sistem transisi baru dengan state abstrak yang jumlah statenya berhingga. Hal ini menunjukkan bahwa apabila terdapat state tak berhingga dapat diabstraksi menjadi state dengan jumlah berhingga.

4.4.2 Menentukan Transisi pada Sistem Transisi Abstrak

Pada langkah sebelumnya diperoleh partisi state, fungsi abstraksi, dan state abstrak. Selanjutnya menentukan transisi pada sistem transisi abstrak. State kongkrit S terhubung dengan state abstrak $\hat{S}$. Apabila pada sistem transisi terdapat transisi dari state s ke state $s^{'}$ yang dinotasikan dengan $s \rightarrow s^{'}$, maka juga terdapat transisi yang besesuaian pada state-state abstrak $\hat{s} \rightarrow_f \hat{s}^{'}$. Untuk dapat mengetahui transisi-transisi yang mungkin, digunakan fungsi abstraksi f seperti yang ada pada (4.1) sedemikian sehingga $f^{-1}(\hat{s}) = \{s : f(s) = \hat{s}\}$.

Jika terdapat transisi dari state s menuju ke state s' pada sistem transisi kongkrit yaitu $s \rightarrow s^{'}$, maka terdapat transisi dari $f(s)$ ke $f(s')$ pada sistem transisi abstrak yaitu dinotasikan dengan $f(s) \rightarrow_f f(s')$.

Contoh 4.4. Berdasarkan Contoh 4.2 dan 4.3, dapat dibuat sistem transisi abstrak. Diketahui terdapat empat state abstrak $\hat{S}_1, \hat{S}_2, \hat{S}_3$, dan $\hat{S}_4$, dimana
$f^{-1}(\hat{S}_1) = \bar{S}_1 = \{(1, [x_1, x_2, x_3])|x_1 > 250, x_2 > 0\}$
$f^{-1}(\hat{S}_2) = \bar{S}_2 = \{(1, [x_1, x_2, x_3])|x_1 > 250, x_2 \leq 0\}$
$f^{-1}(\hat{S}_3) = \bar{S}_3 = \{(1, [x_1, x_2, x_3])|x_1 \leq 250, x_2 > 0\}$
$f^{-1}(\hat{S}_4) = \bar{S}_4 = \{(1, [x_1, x_2, x_3])|x_1 \leq 250, x_2 \leq 0\}$.
Akan ditunjukkan transisi-transisi yang mungkin pada state abstrak sebagai berikut.

1. Transisi-transisi yang mungkin dari $\hat{S}_1$.

   - Membuktikan eksistensi transisi $\hat{S}_1 \rightarrow_f \hat{S}_1$
     Transisi $\hat{S}_1 \rightarrow_f \hat{S}_1$ terjadi apabila terdapat transisi dari state di $\bar{S}_1$ ke state $\bar{S}_1$. State di $\bar{S}_1$ menyatakan terdapat uang lebih dari 250 dan terdapat barang A dalam gudang, yang direpresentasikan dengan $\bar{S}_1 = \{(1, [x_1, x_2, x_3]) | x_1 > 250, x_2 > 0\}$. Sebagai contoh misalkan ambil $s \in S$ dengan $s = (1, [252, 4, 3])$ artinya terdapat uang sejumlah $x_1 = 252$, jumlah barang A yaitu $x_2 = 4$, dan jumlah barang B yaitu $x_3 = 3$, sehingga $s \in \bar{S}_1$. Transisi Jual B dapat di*fire* karena kondisi $x_3 > 0$ terpenuhi, maka state ter*update* dengan $x_1 = 252 + d = 252 + 8 = 260$, $x_2 = 4$, dan $x_3 = 3 - 1 = 2$, sehingga state menjadi $(1, [260, 4, 2])$ yang merupakan anggota dari $\bar{S}_1$. Karena $f(s) = \hat{S}_1$ untuk setiap $s \in \bar{S}_1$, maka terbukti bahwa terdapat transisi $\hat{S}_1 \rightarrow_f \hat{S}_1$.

   - Membuktikan eksistensi transisi $\hat{S}_1 \rightarrow_f \hat{S}_2$
     Transisi $\hat{S}_1 \rightarrow_f \hat{S}_2$ terjadi apabila terdapat transisi dari state di $\bar{S}_1$ ke state $\bar{S}_2$. State di $\bar{S}_1$ menyatakan terdapat uang lebih dari 250 dan terdapat barang A dalam gudang direpresentasikan dengan $\bar{S}_1 = \{(1, [x_1, x_2, x_3]) | x_1 > 250, x_2 > 0\}$. State di $\bar{S}_2$ menyatakan terdapat uang lebih dari 250 tetapi tidak ada barang A dalam gudang direpresentasikan dengan $\bar{S}_2 = \{(1, [x_1, x_2, x_3]) | x_1 > 250, x_2 \leq 0\}$. Sebagai contoh misalkan ambil $s \in S$ dengan $s = (1, [252, 1, 1])$ artinya terdapat uang sejumlah $x_1 = 252$, jumlah barang A yaitu $x_2 = 1$, dan jumlah barang B yaitu $x_3 = 1$, sehingga $s \in \bar{S}_1$. Transisi Jual A dapat di*fire* karena kondisi $x_2 > 0$ terpenuhi, maka state ter*update* dengan $x_1 = 252 + b = 252 + 7 = 259$, $x_2 = 1 - 1 = 0$, dan $x_3 = 1$, sehingga state menjadi $(1, [259, 0, 1])$ yang merupakan anggota dari $\bar{S}_2$. Karena $f(s) = \hat{S}_1$ untuk setiap $s \in \bar{S}_1$ dan $f(s) = \hat{S}_2$ untuk setiap $s \in \bar{S}_2$, maka terbukti bahwa terdapat transisi $\hat{S}_1 \rightarrow_f \hat{S}_2$.

   - Membuktikan eksistensi transisi $\hat{S}_1 \rightarrow_f \hat{S}_3$
     Transisi $\hat{S}_1 \rightarrow_f \hat{S}_3$ terjadi apabila terdapat transisi dari state di $\bar{S}_1$ ke state $\bar{S}_3$. State di $\bar{S}_1$ menyatakan terdapat uang lebih dari 250 dan terdapat barang A dalam gudang direpresentasikan dengan $\bar{S}_1 = \{(1, [x_1, x_2, x_3]) | x_1 > 250, x_2 > 0\}$. State di $\bar{S}_3$ menyatakan

terdapat uang kurang dari 250 tetapi ada barang A dalam gudang direpresentasikan dengan $\bar{S}_3 = \{(1, [x_1, x_2, x_3])|x_1 \leq 250, x_2 > 0\}$. Sebagai contoh misalkan ambil $s \in S$ dengan $s = (1, [252, 1, 1])$ artinya terdapat uang sejumlah $x_1 = 252$, jumlah barang A yaitu $x_2 = 1$, dan jumlah barang B yaitu $x_3 = 1$, sehingga $s \in \bar{S}_1$. Transisi Beli A dapat di*fire* karena kondisi $x_1 > 5$ dan $x_2 + x_3 = 1 + 1 < 40$ terpenuhi, maka state ter*update* dengan $x_1 = 252 - a = 252 - 5 = 247$, $x_2 = 1 + 1 = 2$, dan $x_3 = 1$, sehingga state menjadi $(1, [247, 2, 1])$ yang merupakan anggota dari $\bar{S}_3$. Karena $f(s) = \hat{S}_1$ untuk setiap $s \in \bar{S}_1$ dan $f(s) = \hat{S}_3$ untuk setiap $s \in \bar{S}_3$, maka terbukti bahwa terdapat transisi $\hat{S}_1 \rightarrow_f \hat{S}_3$.

- Membuktikan eksistensi transisi $\hat{S}_1 \rightarrow_f \hat{S}_4$
  Transisi $\hat{S}_1 \rightarrow_f \hat{S}_4$ terjadi apabila terdapat transisi dari state di $\bar{S}_1$ ke state $\bar{S}_4$. State di $\bar{S}_1$ menyatakan terdapat uang lebih dari 250 dan terdapat barang A dalam gudang direpresentasikan dengan $\bar{S}_1 = \{(1, [x_1, x_2, x_3])|x_1 > 250, x_2 > 0\}$. State di $\bar{S}_4$ menyatakan terdapat uang kurang dari 250 dan tidak ada barang A dalam gudang direpresentasikan dengan $\bar{S}_4 = \{(1, [x_1, x_2, x_3])|x_1 \leq 250, x_2 \leq 0\}$. Transisi state di $\bar{S}_1$ ke state $\bar{S}_4$ tidak bisa terjadi karena apabila terdapat uang lebih dari 250 dan terdapat barang A , transisi beli A, beli B, jual A, dan jual B tidak ada yang memenuhi $x_1 \leq 250, x_2 \leq 0$. Jika transisi beli A di*fire*, maka uang berkurang dan jumlah barang A akan bertambah, tetapi tidak memenuhi $x_2 \leq 0$. Jika transisi jual A di*fire*, maka uang bertambah dan jumlah barang A berkurang, tetapi tidak memenuhi $x_1 \leq 250$. Jika transisi beli B di*fire*, maka uang berkurang, jumlah barang A tetap, dan jumlah barang B bertambah, tetapi tidak memenuhi $x_2 \leq 0$. Jika transisi jual B di*fire*, maka uang bertambah, jumlah barang A tetap, dan jumlah barang B berkurang , tetapi tidak memenuhi $x_1 \leq 250, x_2 \leq 0$, sehingga tidak ada state yang memenuhi. Maka tidak terdapat transisi $\hat{S}_1 \rightarrow_f \hat{S}_4$.

  Jadi diperoleh transisi-transisi yang mungkin dari $\hat{S}_1$ adalah $\hat{S}_1 \rightarrow_f \hat{S}_1$, $\hat{S}_1 \rightarrow_f \hat{S}_2$, dan $\hat{S}_1 \rightarrow_f \hat{S}_3$.

2. Transisi-transisi yang mungkin dari $\hat{S}_2$.

   - Membuktikan eksistensi transisi $\hat{S}_2 \rightarrow_f \hat{S}_1$

Transisi $\hat{S}_2 \rightarrow_f \hat{S}_1$ terjadi apabila terdapat transisi dari state di $\bar{S}_2$ ke state $\bar{S}_1$. State di $\bar{S}_2$ menyatakan terdapat uang lebih dari 250 tetapi tidak ada barang A dalam gudang direpresentasikan dengan $\bar{S}_2 = \{(1, [x_1, x_2, x_3])|x_1 > 250, x_2 \leq 0\}$. State di $\bar{S}_1$ menyatakan terdapat uang lebih dari 250 dan terdapat barang A dalam gudang, yang direpresentasikan dengan $\bar{S}_1 = \{(1, [x_1, x_2, x_3])|x_1 > 250, x_2 > 0\}$. Sebagai contoh misalkan ambil $s \in S$ dengan $s = (1, [260, 0, 1])$ artinya terdapat uang sejumlah $x_1 = 260$, jumlah barang A yaitu $x_2 = 0$, dan jumlah barang B yaitu $x_3 = 1$, sehingga $s \in \bar{S}_2$. Transisi beli B dapat di*fire* karena kondisi $x_1 > 5$ dan $x_2 + x_3 = 0 + 1 < 40$ terpenuhi, maka state ter*update* dengan $x_1 = 260 - a = 260 - 5 = 255$, $x_2 = 0 + 1 = 1$, dan $x_3 = 1$, sehingga state menjadi $(1, [255, 1, 1])$ yang merupakan anggota dari $\bar{S}_1$. Karena $f(s) = \hat{S}_2$ untuk setiap $s \in \bar{S}_2$ dan $f(s) = \hat{S}_1$ untuk setiap $s \in \bar{S}_1$, maka terbukti bahwa terdapat transisi $\hat{S}_2 \rightarrow_f \hat{S}_1$.

- Membuktikan eksistensi transisi $\hat{S}_2 \rightarrow_f \hat{S}_2$
  Transisi $\hat{S}_2 \rightarrow_f \hat{S}_2$ terjadi apabila terdapat transisi dari state di $\bar{S}_2$ ke state $\bar{S}_2$. State di $\bar{S}_2$ menyatakan terdapat uang lebih dari 250 tetapi tidak ada barang A dalam gudang direpresentasikan dengan $\bar{S}_2 = \{(1, [x_1, x_2, x_3])|x_1 > 250, x_2 \leq 0\}$. Sebagai contoh misalkan ambil $s \in S$ dengan $s = (1, [252, 0, 3])$ artinya terdapat uang sejumlah $x_1 = 252$, jumlah barang A yaitu $x_2 = 0$, dan jumlah barang B yaitu $x_3 = 3$, sehingga $s \in \bar{S}_2$. Transisi Jual B dapat di*fire* karena kondisi $x_3 > 0$ terpenuhi, maka state ter*update* dengan $x_1 = 252 + d = 252 + 8 = 260$, $x_2 = 0$, dan $x_3 = 3 - 1 = 2$, sehingga state menjadi $(1, [260, 0, 2])$ yang merupakan anggota dari $\bar{S}_2$. Karena $f(s) = \hat{S}_2$ untuk setiap $s \in \bar{S}_2$, maka terbukti bahwa terdapat transisi $\hat{S}_2 \rightarrow_f \hat{S}_2$.

- Membuktikan eksistensi transisi $\hat{S}_2 \rightarrow_f \hat{S}_3$
  Transisi $\hat{S}_2 \rightarrow_f \hat{S}_3$ terjadi apabila terdapat transisi dari state di $\bar{S}_2$ ke state $\bar{S}_3$. State di $\bar{S}_2$ menyatakan terdapat uang lebih dari 250 tetapi tidak ada barang A dalam gudang direpresentasikan dengan $\bar{S}_2 = \{(1, [x_1, x_2, x_3])|x_1 > 250, x_2 \leq 0\}$. State di $\bar{S}_3$ menyatakan terdapat uang kurang dari 250 tetapi ada barang A dalam gudang direpresentasikan dengan $\bar{S}_3 = \{(1, [x_1, x_2, x_3])|x_1 \leq 250, x_2 > 0\}$.

Sebagai contoh misalkan ambil $s \in S$ dengan $s = (1, [252, 0, 1])$ artinya terdapat uang sejumlah $x_1 = 252$, jumlah barang A yaitu $x_2 = 0$, dan jumlah barang B yaitu $x_3 = 1$, sehingga $s \in \bar{S}_2$. Transisi Beli A dapat di*fire* karena kondisi $x_1 > 5$ dan $x_2 + x_3 = 0 + 1 < 40$ terpenuhi, maka state ter*update* dengan $x_1 = 252 - a = 252 - 5 = 247$, $x_2 = 0 + 1 = 1$, dan $x_3 = 1$, sehingga state menjadi $(1, [247, 1, 1])$ yang merupakan anggota dari $\bar{S}_3$. Karena $f(s) = \hat{S}_2$ untuk setiap $s \in \bar{S}_2$ dan $f(s) = \hat{S}_3$ untuk setiap $s \in \bar{S}_3$, maka terbukti bahwa terdapat transisi $\hat{S}_2 \rightarrow_f \hat{S}_3$.

- Membuktikan eksistensi transisi $\hat{S}_2 \rightarrow_f \hat{S}_4$
  Transisi $\hat{S}_2 \rightarrow_f \hat{S}_4$ terjadi apabila terdapat transisi dari state di $\bar{S}_2$ ke state $\bar{S}_4$. State di $\bar{S}_2$ menyatakan terdapat uang lebih dari 250 tetapi tidak ada barang A dalam gudang direpresentasikan dengan $\bar{S}_2 = \{(1, [x_1, x_2, x_3]) | x_1 > 250, x_2 \leq 0\}$. State di $\bar{S}_4$ menyatakan terdapat uang kurang dari 250 dan tidak ada barang A dalam gudang direpresentasikan dengan $\bar{S}_4 = \{(1, [x_1, x_2, x_3]) | x_1 \leq 250, x_2 \leq 0\}$. Sebagai contoh misalkan ambil $s \in S$ dengan $s = (1, [252, 0, 2])$ artinya terdapat uang sejumlah $x_1 = 252$, jumlah barang A yaitu $x_2 = 0$, dan jumlah barang B yaitu $x_3 = 2$, sehingga $s \in \bar{S}_2$. Transisi Beli B dapat di*fire* karena kondisi $x_1 > 6$ dan $x_2 + x_3 = 0 + 2 < 40$ terpenuhi, maka state ter*update* dengan $x_1 = 252 - c = 252 - 6 = 246$, $x_2 = 0$, dan $x_3 = 2 + 1 = 3$, sehingga state menjadi $(1, [246, 0, 3])$ yang merupakan anggota dari $\bar{S}_4$. Karena $f(s) = \hat{S}_2$ untuk setiap $s \in \bar{S}_2$ dan $f(s) = \hat{S}_4$ untuk setiap $s \in \bar{S}_4$, maka terbukti bahwa terdapat transisi $\hat{S}_2 \rightarrow_f \hat{S}_4$.

  Jadi diperoleh transisi-transisi yang mungkin dari $\hat{S}_2$ adalah $\hat{S}_2 \rightarrow_f \hat{S}_1$, $\hat{S}_2 \rightarrow_f \hat{S}_2$, $\hat{S}_2 \rightarrow_f \hat{S}_3$, dan $\hat{S}_2 \rightarrow_f \hat{S}_4$.

3. Transisi-transisi yang mungkin dari $\hat{S}_3$.

   - Membuktikan eksistensi transisi $\hat{S}_3 \rightarrow_f \hat{S}_1$
     Transisi $\hat{S}_3 \rightarrow_f \hat{S}_1$ terjadi apabila terdapat transisi dari state di $\bar{S}_3$ ke state $\bar{S}_1$. State di $\bar{S}_3$ menyatakan terdapat uang kurang dari 250 tetapi ada barang A dalam gudang direpresentasikan dengan $\bar{S}_3 = \{(1, [x_1, x_2, x_3]) | x_1 \leq 250, x_2 > 0\}$. State di $\bar{S}_1$ menyatakan terdapat uang lebih dari 250 dan terdapat barang A dalam gudang, yang direpresentasikan dengan $\bar{S}_1 = \{(1, [x_1, x_2, x_3]) | x_1 > 250, x_2 > 0\}$.

Sebagai contoh misalkan ambil $s \in S$ dengan $s = (1, [248, 2, 3])$ artinya terdapat uang sejumlah $x_1 = 248$, jumlah barang A yaitu $x_2 = 2$, dan jumlah barang B yaitu $x_3 = 3$, sehingga $s \in \bar{S}_3$. Transisi Jual B dapat di*fire* karena kondisi $x_3 > 0$ terpenuhi, maka state ter*update* dengan $x_1 = 248 + d = 248 + 8 = 256$, $x_2 = 2$, dan $x_3 = 3 - 1 = 2$, sehingga state menjadi $(1, [256, 2, 2])$ yang merupakan anggota dari $\bar{S}_1$. Karena $f(s) = \hat{S}_3$ untuk setiap $s \in \bar{S}_3$ dan $f(s) = \hat{S}_1$ untuk setiap $s \in \bar{S}_1$, maka terbukti bahwa terdapat transisi $\hat{S}_3 \rightarrow_f \hat{S}_1$.

- Membuktikan eksistensi transisi $\hat{S}_3 \rightarrow_f \hat{S}_2$
  Transisi $\hat{S}_3 \rightarrow_f \hat{S}_2$ terjadi apabila terdapat transisi dari state di $\bar{S}_3$ ke state $\bar{S}_2$. State di $\bar{S}_3$ menyatakan terdapat uang kurang dari 250 tetapi ada barang A dalam gudang direpresentasikan dengan $\bar{S}_3 = \{(1, [x_1, x_2, x_3]) | x_1 \leq 250, x_2 > 0\}$. State di $\bar{S}_2$ menyatakan terdapat uang lebih dari 250 tetapi tidak ada barang A dalam gudang direpresentasikan dengan $\bar{S}_2 = \{(1, [x_1, x_2, x_3]) | x_1 > 250, x_2 \leq 0\}$. Sebagai contoh misalkan ambil $s \in S$ dengan $s = (1, [248, 1, 1])$ artinya terdapat uang sejumlah $x_1 = 248$, jumlah barang A yaitu $x_2 = 1$, dan jumlah barang B yaitu $x_3 = 1$, sehingga $s \in \bar{S}_3$. Transisi Jual A dapat di*fire* karena kondisi $x_2 > 0$ terpenuhi, maka state ter*update* dengan $x_1 = 248 + b = 248 + 7 = 255$, $x_2 = 1 - 1 = 0$, dan $x_3 = 1$, sehingga state menjadi $(1, [255, 0, 1])$ yang merupakan anggota dari $\bar{S}_2$. Karena $f(s) = \hat{S}_3$ untuk setiap $s \in \bar{S}_3$ dan $f(s) = \hat{S}_2$ untuk setiap $s \in \bar{S}_2$, maka terbukti bahwa terdapat transisi $\hat{S}_3 \rightarrow_f \hat{S}_2$.

- Membuktikan eksistensi transisi $\hat{S}_3 \rightarrow_f \hat{S}_3$
  Transisi $\hat{S}_3 \rightarrow_f \hat{S}_3$ terjadi apabila terdapat transisi dari state di $\bar{S}_3$ ke state $\bar{S}_3$. State di $\bar{S}_3$ menyatakan terdapat uang kurang dari 250 tetapi ada barang A dalam gudang direpresentasikan dengan $\bar{S}_3 = \{(1, [x_1, x_2, x_3]) | x_1 \leq 250, x_2 > 0\}$. Sebagai contoh misalkan ambil $s \in S$ dengan $s = (1, [245, 1, 2])$ artinya terdapat uang sejumlah $x_1 = 245$, jumlah barang A yaitu $x_2 = 1$, dan jumlah barang B yaitu $x_3 = 2$, sehingga $s \in \bar{S}_3$. Transisi Beli B dapat di*fire* karena kondisi $x_1 > 6$ dan $x_2 + x_3 = 1 + 2 < 40$ terpenuhi, maka state ter*update* dengan $x_1 = 245 - c = 245 - 6 = 239$, $x_2 = 1$, dan $x_3 = 2 + 1 = 3$, sehingga state menjadi $(1, [239, 1, 3])$ yang merupakan anggota dari $\bar{S}_3$.

Karena $f(s) = \hat{S}_3$ untuk setiap $s \in \bar{S}_3$, maka terbukti bahwa terdapat transisi $\hat{S}_3 \rightarrow_f \hat{S}_3$.

- Membuktikan eksistensi transisi $\hat{S}_3 \rightarrow_f \hat{S}_4$
  Transisi $\hat{S}_3 \rightarrow_f \hat{S}_4$ terjadi apabila terdapat transisi dari state di $\bar{S}_3$ ke state $\bar{S}_4$. State di $\bar{S}_3$ menyatakan terdapat uang kurang dari 250 tetapi ada barang A dalam gudang direpresentasikan dengan $\bar{S}_3 = \{(1, [x_1, x_2, x_3]) | x_1 \leq 250, x_2 > 0\}$. State di $\bar{S}_4$ menyatakan terdapat uang kurang dari 250 dan tidak ada barang A dalam gudang direpresentasikan dengan $\bar{S}_4 = \{(1, [x_1, x_2, x_3]) | x_1 \leq 250, x_2 \leq 0\}$. Sebagai contoh misalkan ambil $s \in S$ dengan $s = (1, [240, 1, 1])$ artinya terdapat uang sejumlah $x_1 = 240$, jumlah barang A yaitu $x_2 = 1$, dan jumlah barang B yaitu $x_3 = 1$, sehingga $s \in \bar{S}_3$. Transisi Jual A dapat di*fire* karena kondisi $x_2 > 0$ terpenuhi, maka state ter*update* dengan $x_1 = 240 + b = 240 + 7 = 247$, $x_2 = 1 - 1 = 0$, dan $x_3 = 1$, sehingga state menjadi $(1, [247, 0, 1])$ yang merupakan anggota dari $\bar{S}_4$. Karena $f(s) = \hat{S}_3$ untuk setiap $s \in \bar{S}_3$ dan $f(s) = \hat{S}_4$ untuk setiap $s \in \bar{S}_4$, maka terbukti bahwa terdapat transisi $\hat{S}_3 \rightarrow_f \hat{S}_4$.

  Jadi diperoleh transisi-transisi yang mungkin dari $\hat{S}_3$ adalah $\hat{S}_3 \rightarrow_f \hat{S}_1$, $\hat{S}_3 \rightarrow_f \hat{S}_2$, $\hat{S}_3 \rightarrow_f \hat{S}_3$, dan $\hat{S}_3 \rightarrow_f \hat{S}_4$.

4. Transisi-transisi yang mungkin dari $\hat{S}_4$.

   - Membuktikan eksistensi transisi $\hat{S}_4 \rightarrow_f \hat{S}_1$
     Transisi $\hat{S}_4 \rightarrow_f \hat{S}_1$ terjadi apabila terdapat transisi dari state di $\bar{S}_4$ ke state $\bar{S}_1$. State di $\bar{S}_4$ menyatakan terdapat uang kurang dari 250 dan tidak ada barang A dalam gudang direpresentasikan dengan $\bar{S}_4 = \{(1, [x_1, x_2, x_3]) | x_1 \leq 250, x_2 \leq 0\}$. State di $\bar{S}_1$ menyatakan terdapat uang lebih dari 250 dan terdapat barang A dalam gudang direpresentasikan dengan $\bar{S}_1 = \{(1, [x_1, x_2, x_3]) | x_1 > 250, x_2 > 0\}$. Transisi state di $\bar{S}_4$ ke state $\bar{S}_1$ tidak bisa terjadi karena apabila terdapat uang kurang dari 250 dan tidak ada barang A, transisi beli A, beli B, jual A, dan jual B tidak terjadi, sehingga tidak ada yang memenuhi $x_1 > 250, x_2 > 0$. Jika transisi beli A di*fire*, maka uang berkurang dan jumlah barang A akan bertambah, tetapi tidak memenuhi $x_1 > 250$. Transisi jual A tidak bisa di*fire* karena tidak memenuhi kondisi $x_2 > 0$.

Jika transisi beli B di*fire*, maka uang berkurang, jumlah barang A tetap, dan jumlah barang B bertambah, tetapi tidak memenuhi $x_1 > 250$. Jika transisi jual B di*fire*, maka uang bertambah, jumlah barang A tetap,dan jumlah barang B berkurang , tetapi tidak memenuhi $x_2 > 0$, sehingga tidak ada state yang memenuhi. Maka tidak terdapat transisi $\hat{S}_4 \rightarrow_f \hat{S}_1$.

- Membuktikan eksistensi transisi $\hat{S}_4 \rightarrow_f \hat{S}_2$
  Transisi $\hat{S}_4 \rightarrow_f \hat{S}_2$ terjadi apabila terdapat transisi dari state di $\bar{S}_4$ ke state $\bar{S}_2$. State di $\bar{S}_4$ menyatakan terdapat uang kurang dari 250 dan tidak ada barang A dalam gudang direpresentasikan dengan $\bar{S}_4 = \{(1, [x_1, x_2, x_3]) | x_1 \leq 250, x_2 \leq 0\}$. State di $\bar{S}_2$ menyatakan terdapat uang lebih dari 250 tetapi tidak ada barang A dalam gudang direpresentasikan dengan $\bar{S}_2 = \{(1, [x_1, x_2, x_3]) | x_1 > 250, x_2 \leq 0\}$. Sebagai contoh misalkan ambil $s \in S$ dengan $s = (1, [248, 0, 3])$ artinya terdapat uang sejumlah $x_1 = 248$, jumlah barang A yaitu $x_2 = 0$, dan jumlah barang B yaitu $x_3 = 3$, sehingga $s \in \bar{S}_4$. Transisi Jual B dapat di*fire* karena kondisi $x_3 > 0$ terpenuhi, maka state ter*update* dengan $x_1 = 248 + d = 248 + 8 = 256$, $x_2 = 0$, dan $x_3 = 3 - 1 = 2$, sehingga state menjadi $(1, [256, 0, 2])$ yang merupakan anggota dari $\bar{S}_2$. Karena $f(s) = \hat{S}_4$ untuk setiap $s \in \bar{S}_4$ dan $f(s) = \hat{S}_2$ untuk setiap $s \in \bar{S}_2$, maka terbukti bahwa terdapat transisi $\hat{S}_4 \rightarrow_f \hat{S}_2$.

- Membuktikan eksistensi transisi $\hat{S}_4 \rightarrow_f \hat{S}_3$
  Transisi $\hat{S}_4 \rightarrow_f \hat{S}_3$ terjadi apabila terdapat transisi dari state di $\bar{S}_4$ ke state $\bar{S}_3$. State di $\bar{S}_4$ menyatakan terdapat uang kurang dari 250 dan tidak ada barang A dalam gudang direpresentasikan dengan $\bar{S}_4 = \{(1, [x_1, x_2, x_3]) | x_1 \leq 250, x_2 \leq 0\}$. State di $\bar{S}_3$ menyatakan terdapat uang kurang dari 250 tetapi ada barang A dalam gudang direpresentasikan dengan $\bar{S}_3 = \{(1, [x_1, x_2, x_3]) | x_1 \leq 250, x_2 > 0\}$. Sebagai contoh misalkan ambil $s \in S$ dengan $s = (1, [245, 0, 1])$ artinya terdapat uang sejumlah $x_1 = 245$, jumlah barang A yaitu $x_2 = 0$, dan jumlah barang B yaitu $x_3 = 1$, sehingga $s \in \bar{S}_4$. Transisi Beli A dapat di*fire* karena kondisi $x_1 > 5$ dan $x_2 + x_3 = 0 + 1 < 40$ terpenuhi, maka state ter*update* dengan $x_1 = 245 - a = 245 - 5 = 240$, $x_2 = 0 + 1 = 1$, dan $x_3 = 1$, sehingga state menjadi $(1, [240, 1, 1])$ yang merupakan anggota dari $\bar{S}_3$. Karena $f(s) = \hat{S}_4$ untuk setiap $s \in \bar{S}_4$ dan $f(s) = \hat{S}_3$

untuk setiap $s \in \bar{S}_3$, maka terbukti bahwa terdapat transisi $\hat{S}_4 \rightarrow_f \hat{S}_3$.

- Membuktikan eksistensi transisi $\hat{S}_4 \rightarrow_f \hat{S}_4$
  Transisi $\hat{S}_4 \rightarrow_f \hat{S}_4$ terjadi apabila terdapat transisi dari state di $\bar{S}_4$ ke state $\bar{S}_4$. State di $\bar{S}_4$ menyatakan terdapat uang kurang dari 250 dan tidak ada barang A dalam gudang direpresentasikan dengan $\bar{S}_4 = \{(1, [x_1, x_2, x_3]) | x_1 \leq 250, x_2 \leq 0\}$. Sebagai contoh misalkan ambil $s \in S$ dengan $s = (1, [245, 0, 1])$ artinya terdapat uang sejumlah $x_1 = 245$, jumlah barang A yaitu $x_2 = 0$, dan jumlah barang B yaitu $x_3 = 1$, sehingga $s \in \bar{S}_4$. Transisi Beli B dapat di*fire* karena kondisi $x_1 > 6$ dan $x_2 + x_3 = 0 + 1 < 40$ terpenuhi, maka state ter*update* dengan $x_1 = 245 - c = 245 - 6 = 239$, $x_2 = 0$, dan $x_3 = 1 + 1 = 2$, sehingga state menjadi $(1, [239, 0, 2])$ yang merupakan anggota dari $\bar{S}_4$. Karena $f(s) = \hat{S}_4$ untuk setiap $s \in \bar{S}_4$, maka terbukti bahwa terdapat transisi $\hat{S}_4 \rightarrow_f \hat{S}_4$.

  Jadi diperoleh transisi-transisi yang mungkin dari $\hat{S}_4$ adalah $\hat{S}_4 \rightarrow_f \hat{S}_2$, $\hat{S}_4 \rightarrow_f \hat{S}_3$, dan $\hat{S}_4 \rightarrow_f \hat{S}_4$.

Berdasarkan kemungkinan transisi-transisi yang terjadi pada state abstrak, diperoleh sistem transisi abstrak pada Gambar 4.5 merupakan gambar sistem transisi abstrak dari sistem transisi pada Gambar 4.3, terdapat empat state abstrak. Dari jumlah state tak berhingga sebelumnya pada sistem transisi pada Gambar 4.3 dapat diperoleh sistem transisi yang berhingga.

Selanjutnya pelabelan sistem transisi abstrak, dengan diketahui AP dari Contoh 4.3 yaitu $AP = \{g_1, g_2\}$ dengan himpunan state yang memenuhi g_1 adalah $\{x | x_1 > 250\}$ sedangkan himpunan state yang memenuhi g_2 adalah $\{x | x_2 > 0\}$, dimana g_1 berarti mendapat keuntungan dan g_2 berarti tersedianya barang A dalam gudang.

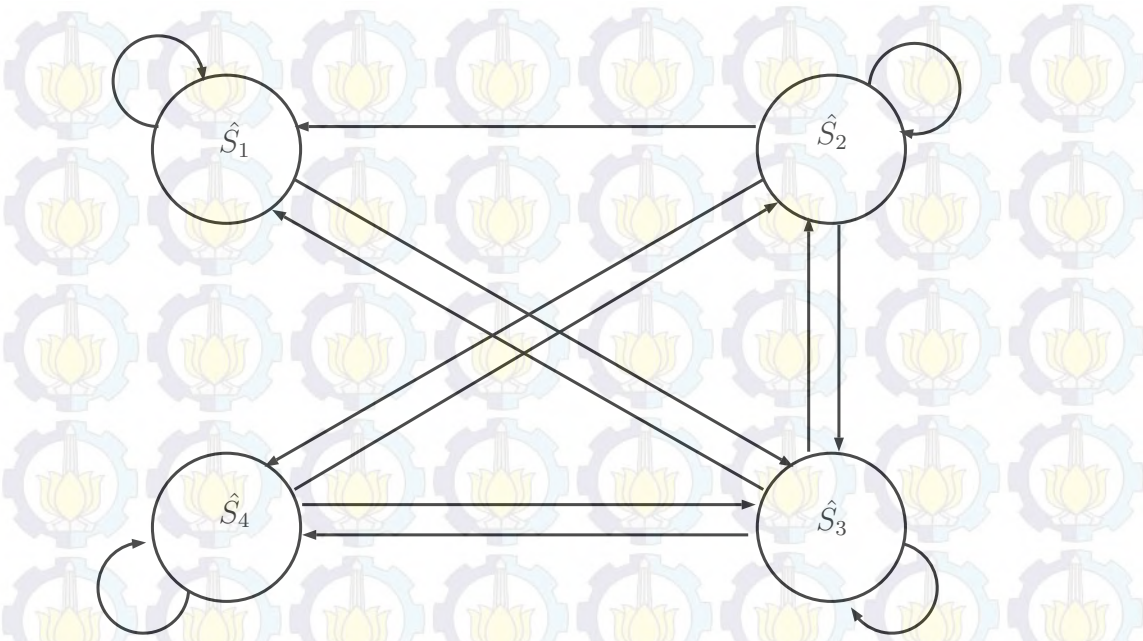
Pada sistem transisi abstrak di atas, diketahui terdapat transisi dari setiap state abstrak ke state abstrak lainnya. Label dari masing-masing state abstrak diatas adalah

$$Lb(\hat{S}_1) = \{g_1, g_2\},$$

$$Lb(\hat{S}_2) = \{g_1\},$$

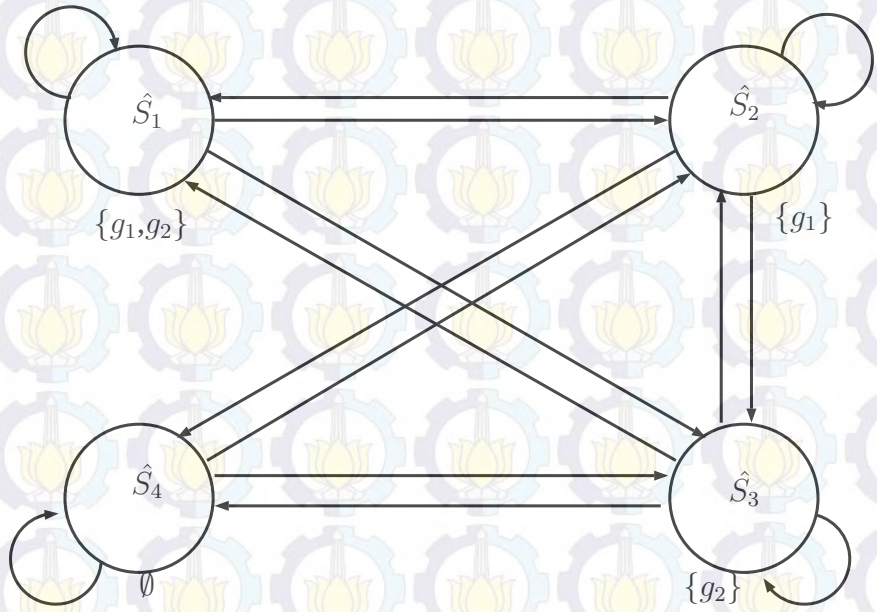
$$Lb(\hat{S}_3) = \{g_2\},$$

$$Lb(\hat{S}_4) = \emptyset.$$



Gambar 4.5: Sistem Transisi Abstrak PNZ Sistem Inventori

Berdasarkan label tersebut, $Lb(\hat{S}_1) = \{g_1, g_2\}$ berarti bahwa state abstrak $\hat{S}_1$ beranggotakan state-state yang ada pada partisi $\bar{S}_1 = \{(1, [x_1, x_2, x_3])|x_1 > 250, x_2 > 0\}$, dimana state-state tersebut menyatakan terdapat uang lebih dari 250 dan terdapat barang A dalam gudang. Berarti bahwa di state abstrak $\hat{S}_1$ Agen mendapat keuntungan dan memiliki barang A di gudang. Pada $Lb(\hat{S}_2) = \{g_1\}$ berarti bahwa state abstrak $\hat{S}_2$ beranggotakan state-state yang ada pada partisi $\bar{S}_2 = \{(1, [x_1, x_2, x_3])|x_1 > 250, x_2 \leq 0\}$, dimana state-state tersebut menyatakan terdapat uang lebih dari 250 tetapi tidak ada barang A dalam gudang. Berarti bahwa Agen mendapat keuntungan. Pada $Lb(\hat{S}_3) = \{g_2\}$ berarti bahwa state abstrak $\hat{S}_3$ beranggotakan state-state yang ada pada partisi $\bar{S}_3 = \{(1, [x_1, x_2, x_3])|x_1 \leq 250, x_2 > 0\}$, dimana state-state tersebut menyatakan terdapat uang kurang dari 250 tetapi ada barang A dalam gudang. Berarti bahwa Agen tidak untung tetapi memiliki barang A dalam gudang. Pada $Lb(\hat{S}_4) = \emptyset$ berarti bahwa state abstrak $\hat{S}_4$ beranggotakan state-state yang ada pada partisi $\bar{S}_4 = \{(1, [x_1, x_2, x_3])|x_1 \leq 250, x_2 \leq 0\}$, dimana state-state tersebut terdapat uang kurang dari 250 dan tidak ada barang A dalam gudang. Berarti bahwa agen tidak untung dan tidak memiliki barang A dalam gudang. Hal ini juga menunjukkan bahwa terdapat transisi dari state ke state diantara daerah partisi pada sistem transisi kongkrit. Tetapi, pada sistem transisi abstrak kita tidak dapat mengetahui dengan pasti state-state dan transisinya.



Gambar 4.6: Pelabelan Sistem Transisi Abstrak PNZ Sistem Inventori

4.5 Spesifikasi

Tujuan dari verifikasi suatu model sistem adalah untuk mengetahui apakah model yang dimiliki telah sesuai dan memenuhi spesifikasi yang diinginkan. Spesifikasi formal dinyatakan dalam notasi matematika yang didefinisikan, sintaksis dan semantik. Spesifikasi bernilai benar (*true*) atau salah (*false*). Spesifikasi dibuat berdasarkan keinginan agar model dapat merepresentasikan sistem sesuai dengan kebutuhan, sehingga dapat meminimalisir kesalahan sistem.

Model dari sistem memenuhi spesifikasi apabila semua path (baraisan dari state-state) memenuhi spesifikasi. Pada penelitian ini, model PNZ sistem inventori dapat dibentuk beberapa spesifikasi yang akan diberikan dalam contoh berikut ini.

Contoh 4.5. Diberikan spesifikasi untuk verifikasi formal PNZ. Diketahui suatu sistem transisi pada Contoh 4.3 dengan $AP = \{g_1, g_2\}$ dimana himpunan state yang memenuhi g_1 adalah $\{x|x_1 > 250\}$ sedangkan himpunan state yang memenuhi g_2 adalah $\{x|x_2 > 0\}$, dimana g_1 berarti mendapat keuntungan dan g_2 berarti tersedianya barang A dalam gudang, maka beberapa spesifikasi yang dapat digunakan adalah:

- Pada transaksi berikutnya Agen selalu mendapatkan keuntungan, formulanya nya adalah $\bigcirc(\Box\{x|x_1 > 250\})$ atau $\bigcirc(\Box g_1)$.
- Pada suatu waktu barang A selalu ada dalam gudang, formulanya adalah $\diamondsuit(\Box\{x|x_2 > 0\})$ atau $\diamondsuit(\Box g_2)$.

- Pada suatu waktu yang sama Agen memperoleh keuntungan dan tersedianya barang A dalam gudang , formulanya adalah $\diamondsuit(\{x|x_1 > 250\} \wedge \{x|x_2 > 0\})$ atau $\diamondsuit(g_1 \wedge g_2)$.
- Pada saat Agen tidak mendapat keuntungan tetapi memiliki barang A dalam gudang, maka suatu saat Agen dapat memperoleh keuntungan, formulanya adalah $\diamondsuit((\{x|x_1 \leq 250\} \wedge \{x|x_2 > 0\}) \rightarrow \{x|x_1 > 250\})$ atau $\diamondsuit((\leftharpoondown g_1 \wedge g_2) \rightarrow \bigcirc g_1)$.

MODULE main
VAR
state : $\{s1, s2, s3, s4\}$;
ASSIGN
init(state):=s1;
next(state) := case
state = s4 : $\{s2, s3, s4\}$;
state = s3 : $\{s1, s2, s3, s4\}$;
state = s2 : $\{s1, s2, s3, s4\}$;
state = s1 : $\{s1, s2, s3\}$
TRUE : $\{s1, s2, s3, s4\}$;
esac;
DEFINE
g1 := $state = s1 \mid state = s2$;
g2 := $state = s1 \mid state = s3$;
LTLSPEC X (G g1);
LTLSPEC F (G g2);
LTLSPEC F (g1 dan g2);
LTLSPEC F (! g1 dan g2 $->$ X g1);

Gambar 4.7: Sistem Transisi Abstrak dengan Bahasa NuSMV

Sistem transisi abstrak pada Gambar 4.6 diterjemahkan ke dalam bahasa NuSMV terdapat pada Gambar 4.7. Bagian VAR untuk mendefinisikan state-state, sedangkan bagian ASSIGN untuk mendeskripsikan *initial* state dan transisi dari masing-masing state.

4.6 Implementasi dengan NuSMV

Verifikasi formal dapat dilakukan menggunakan *software* model *checker*, dengan mendefinisikannya terlebih dahulu dalam bahasa NuSMV. Setelah didefin-

isikan Pada NuSMV, spesifikasi dapat dicek dan dievaluasi. Ketika spesifikasi salah atau tidak terpenuhi, NuSMV memberikan *counterexample*. Pada NuSMV state abstrak dapat dituliskan seperti state biasa agar lebih sederhana dalam penulisan, seperti $\hat{S}$ menjadi S.

Contoh 4.6. Didefinisikan sistem transisi dalam bahasa NuSMV pada Gambar 4.7 berdasarkan sistem transisi abstrak pada Gambar 4.6 dan spesifikasi pada Contoh 4.5. Berikut akan diberikan hasil-hasil verifikasi dengan *initial* state yang berbeda-beda.

- Verifikasi dengan *initial* state $\hat{S}_1$
  Apabila *initial* state adalah $\hat{S}_1$ dengan $L_b = \{g_1, g_2\}$ berarti bahwa awalnya Agen telah memiliki uang lebih dari 250 yaitu $\{x|x_1 > 250\}$ dan terdapat barang A dalam gudang yaitu $\{x|x_2 > 0\}$. Akan ditunjukkan apakah sistem memenuhi spesifikasi-spesifikasi dengan hasil verifikasi pada NuSMV diberikan pada Gambar 4.8.

```
-- specification  X ( G g1)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
    state = s1
    g2 = TRUE
    g1 = TRUE
  -> State: 1.2 <-
    state = s3
    g1 = FALSE
  -- Loop starts here
  -> State: 1.3 <-
    state = s1
    g1 = TRUE
  -> State: 1.4 <-
    state = s2
    g2 = FALSE
  -> State: 1.5 <-
    state = s1
    g2 = TRUE
-- specification  F ( G g2)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -- Loop starts here
  -> State: 2.1 <-
    state = s1
    g2 = TRUE
    g1 = TRUE
  -> State: 2.2 <-
    state = s2
    g2 = FALSE
  -> State: 2.3 <-
    state = s1
    g2 = TRUE
-- specification  F (g1 & g2)  is true
-- specification  F ((!g1 & g2) ->  X g1)  is true
```

Gambar 4.8: Hasil Verifikasi dengan *Initial* State $\hat{S}_1$

1. Spesifikasi $\bigcirc(\Box g_1)$ yang berarti bahwa pada transaksi berikutnya Agen selalu mendapatkan keuntungan, spesifikasi tidak dipenuhi dengan *counterexample* yaitu pada state 1.2 state S_3 dituliskan $g_1 = false$ karena pada S_3

yang berarti $\hat{S}_3$ memiliki label $\{g2\}$ menyatakan bahwa pada state tersebut $\{x_1 \leq 250, x_2 > 0\}$ sedangkan yang diinginkan adalah state berikutnya selalu $\{x|x_1 > 250\}$, maka spesifikasi tidak dipenuhi.

2. Spesifikasi $\diamondsuit(\Box g_2)$ yang berarti bahwa pada suatu waktu barang A selalu ada dalam gudang, spesifikasi tidak dipenuhi dengan *counterexample* yaitu terjadinya *looping* dari state $S_1 - S_2 - S_1 - ...$ sehingga terdapat path tak berhingga dan setiap state nya tidak pernah benar. Hal ini karena pada state S_2 yang berarti $\hat{S}_2$ memiliki label $\{g1\}$ menyatakan bahwa pada state tersebut $\{x_1 > 250, x_2 \leq 0\}$ sedangkan yang diinginkan adalah akhirnya state selalu $\{x|x_2 > 0\}$, maka spesifikasi tidak dipenuhi.

3. Spesifikasi $\diamondsuit(g_1 \wedge g_2)$ yang berarti bahwa pada suatu waktu akhirnya Agen memperoleh keuntungan dan tersedianya barang A dalam gudang , terpenuhi.

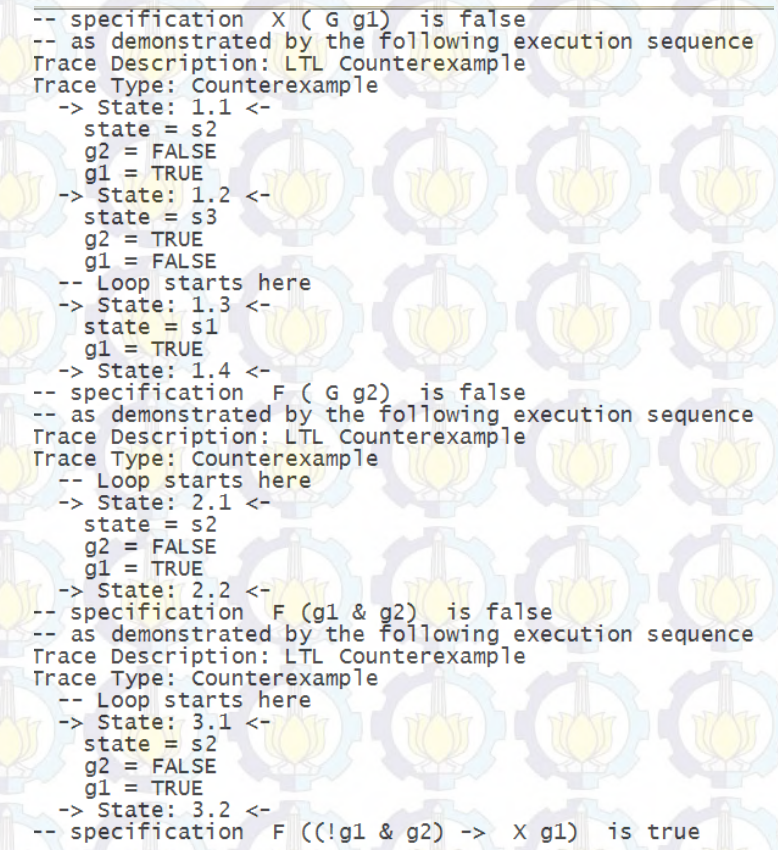
4. Spesifikasi $\diamondsuit((\leftharpoondown g_1 \wedge g_2) \rightarrow \bigcirc g_1)$ yang berarti pada saat Agen tidak mendapat keuntungan tetapi memiliki barang A dalam gudang, maka akhirnya suatu saat Agen dapat memperoleh keuntungan, spesifikasi terpenuhi.

Jadi dengan *initial* state $\hat{S}_1$, spesifikasi yang dipenuhi adalah spesifikasi $\diamondsuit(g_1 \wedge g_2)$ dan spesifikasi $\diamondsuit((\leftharpoondown g_1 \wedge g_2) \rightarrow \bigcirc g_1)$.

- Verifikasi dengan *initial* state $\hat{S}_2$
  Apabila *initial* state adalah $\hat{S}_2$ dengan $L_b = \{g_1\}$ berarti bahwa awalnya Agen memiliki uang lebih dari 250 yaitu $\{x|x_1 > 250\}$ dan tidak terdapat barang A dalam gudang yaitu $\{x|x_2 \leq 0\}$. Akan ditunjukkan apakah sistem memenuhi spesifikasi-spesifikasi dengan hasil verifikasi pada NuSMV diberikan pada Gambar 4.9.

1. Spesifikasi $\bigcirc(\Box g_1)$ yang berarti bahwa pada transaksi berikutnya Agen selalu mendapatkan keuntungan, spesifikasi tidak dipenuhi dengan *counterexample* yaitu pada state 1.2 state S_3 dituliskan $g_1 = false$ karena pada S_3 yang berarti $\hat{S}_3$ memiliki label $\{g2\}$ menyatakan bahwa pada state tersebut $\{x_1 \leq 250, x_2 > 0\}$ sedangkan yang diinginkan adalah state berikutnya selalu $\{x|x_1 > 250\}$, maka spesifikasi tidak dipenuhi.

2. Spesifikasi $\diamondsuit(\Box g_2)$ menyatakan suatu saat $(\Box g_2)$ selalu benar yang berarti bahwa pada suatu waktu barang A selalu ada dalam gudang, spesifikasi



Gambar 4.9: Hasil Verifikasi dengan *Initial* State $\hat{S}_2$

tidak dipenuhi dengan *counterexample* yaitu terjadinya *looping* di state $S_2 - S_2 - S_2 - ...$ sehingga terdapat path tak berhingga dan setiap state nya tidak pernah benar. Hal ini karena pada state S_2 yang berarti $\hat{S}_2$ memiliki label $\{g1\}$ menyatakan bahwa pada state tersebut $\{x_1 > 250, x_2 \leq 0\}$ sedangkan yang diinginkan adalah akhirnya state selalu $\{x | x_2 > 0\}$, maka spesifikasi tidak dipenuhi.

3. Spesifikasi $\diamondsuit(g_1 \wedge g_2)$ menyatakan suatu saat $(g_1 \wedge g_2)$ selalu benar yang berarti bahwa pada suatu waktu akhirnya Agen memperoleh keuntungan dan tersedianya barang A dalam gudang , spesifikasi tidak dipenuhi dengan *counterexample* yaitu terjadinya *looping* di state $S_2 - S_2 - S_2 - ...$ sehingga terdapat path tak berhingga dan setiap state nya tidak pernah benar. Hal ini karena pada state S_2 yang berarti $\hat{S}_2$ memiliki label $\{g1\}$ menyatakan bahwa pada state tersebut $\{x_1 > 250, x_2 \leq 0\}$ sedangkan yang diinginkan adalah akhirnya state $\{x_1 > 250, x_2 > 0\}$, maka spesifikasi tidak dipenuhi.

4. Spesifikasi $\diamondsuit((\leftharpoondown g_1 \wedge g_2) \rightarrow \bigcirc g_1)$ yang berarti pada saat Agen

tidak mendapat keuntungan tetapi memiliki barang A dalam gudang, maka akhirnya suatu saat Agen dapat memperoleh keuntungan, spesifikasi terpenuhi.

Jadi dengan *initial* state $\hat{S}_2$, spesifikasi yang dipenuhi adalah spesifikasi $\diamondsuit((\leftharpoondown g_1 \wedge g_2) \rightarrow \bigcirc g_1)$.

- Verifikasi dengan *initial* state $\hat{S}_3$
  Apabila *initial* state adalah $\hat{S}_3$ dengan $L_b = \{g_2\}$ berarti bahwa Agen memiliki uang kurang dari 250 yaitu $\{x|x_1 \leq 250\}$ dan mempunyai barang A dalam gudang yaitu $\{x|x_2 > 0\}$. Akan ditunjukkan apakah sistem memenuhi spesifikasi-spesifikasi dengan hasil verifikasi pada NuSMV diberikan pada Gambar 4.10.

1. Spesifikasi $\bigcirc(\Box g_1)$ yang berarti bahwa pada transaksi berikutnya Agen selalu mendapatkan keuntungan, spesifikasi tidak dipenuhi dengan *counterexample* yaitu terjadinya *looping* di state $S_3 - S_3 - S_3 - ...$ sehingga terdapat path tak berhingga dan state berikutnya tidak pernah ditemukan $(\Box g_1)$ berarti g_1 tidak pernah benar. Hal ini karena pada state S_3 yang berarti $\hat{S}_3$ memiliki label $\{g2\}$ menyatakan bahwa pada state tersebut $\{x_1 \leq 250, x_2 > 0\}$ sedangkan yang diinginkan adalah state berikutnya selalu $\{x|x_1 > 250\}$, maka spesifikasi tidak dipenuhi.

2. Spesifikasi $\diamondsuit(\Box g_2)$ yang berarti bahwa pada suatu waktu barang A selalu ada dalam gudang, spesifikasi tidak dipenuhi dengan *counterexample* yaitu terjadinya *looping* di state $S_3 - S_2 - S_3 - ...$ sehingga terdapat path tak berhingga dan setiap state nya tidak pernah benar. Hal ini karena pada state S_2 yang berarti $\hat{S}_2$ memiliki label $\{g1\}$ menyatakan bahwa pada state tersebut $\{x_1 > 250, x_2 \leq 0\}$ sedangkan yang diinginkan adalah akhirnya state selalu $\{x|x_2 > 0\}$, maka spesifikasi tidak dipenuhi.

3. Spesifikasi $\diamondsuit(g_1 \wedge g_2)$ menyatakan suatu saat $(g_1 \wedge g_2)$ selalu benar yang berarti bahwa pada suatu waktu akhirnya Agen memperoleh keuntungan dan tersedianya barang A dalam gudang , spesifikasi tidak dipenuhi dengan *counterexample* yaitu terjadinya *looping* di state $S_3 - S_3 - S_3 - ...$ sehingga terdapat path tak berhingga dan setiap state nya tidak pernah benar. Hal ini karena pada state S_3 yang berarti $\hat{S}_3$ memiliki label $\{g2\}$ menyatakan bahwa pada state tersebut $\{x_1 \leq 250, x_2 > 0\}$ sedangkan yang diinginkan adalah akhirnya state $\{x_1 > 250, x_2 > 0\}$, maka spesifikasi tidak dipenuhi.

```
-- specification  X ( G g1)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -- Loop starts here
  -> State: 1.1 <-
    state = s3
    g2 = TRUE
    g1 = FALSE
  -> State: 1.2 <-
-- specification  F ( G g2)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -- Loop starts here
  -> State: 2.1 <-
    state = s3
    g2 = TRUE
    g1 = FALSE
  -> State: 2.2 <-
    state = s2
    g2 = FALSE
    g1 = TRUE
  -> State: 2.3 <-
    state = s3
    g2 = TRUE
    g1 = FALSE
-- specification  F (g1 & g2)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -- Loop starts here
  -> State: 3.1 <-
    state = s3
    g2 = TRUE
    g1 = FALSE
  -> State: 3.2 <-
-- specification  F ((!g1 & g2) ->  X g1)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -- Loop starts here
  -> State: 4.1 <-
    state = s3
    g2 = TRUE
    g1 = FALSE
  -> State: 4.2 <-
```

Gambar 4.10: Hasil Verifikasi dengan *Initial* State $\hat{S}_3$

4. Spesifikasi $\Diamond((\leftharpoondown \ g_1 \wedge g_2) \ \rightarrow \ \bigcirc g_1)$ yang berarti pada saat Agen tidak mendapat keuntungan tetapi memiliki barang A dalam gudang, maka akhirnya suatu saat Agen dapat memperoleh keuntungan, spesifikasi tidak dipenuhi dengan *counterexample* yaitu terjadinya *looping* di state $S_3 - S_3 - S_3 - ...$ sehingga terdapat path tak berhingga dan setiap state nya tidak pernah benar. Hal ini karena pada state S_3 yang berarti $\hat{S}_3$ memiliki label $\{g2\}$ menyatakan bahwa pada state tersebut $\{x_1 \leq 250, x_2 > 0\}$ sedangkan yang diinginkan adalah akhirnya state $\{x_1 > 250\}$ tidak pernah bisa ditemukan g_1, maka spesifikasi tidak dipenuhi.

Jadi dengan *initial* state $\hat{S}_3$, tidak ada spesifikasi yang dipenuhi.

- Verifikasi dengan *initial* state $\hat{S}_4$
  Apabila *initial* state adalah $\hat{S}_4$ dengan $L_b = \emptyset$ berarti bahwa awalnya Agen memiliki uang kurang dari 250 yaitu $\{x|x_1 \leq 250\}$ dan tidak terdapat barang A dalam gudang yaitu $\{x|x_2 \leq 0\}$. Akan ditunjukkan apakah sistem memenuhi spesifikasi-spesifikasi dengan hasil verifikasi pada NuSMV diberikan pada Gambar 4.11.

```
-- specification  X ( G g1)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -- Loop starts here
  -> State: 1.1 <-
    state = s4
    g2 = FALSE
    g1 = FALSE
  -> State: 1.2 <-
-- specification  F ( G g2)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -- Loop starts here
  -> State: 2.1 <-
    state = s4
    g2 = FALSE
    g1 = FALSE
  -> State: 2.2 <-
-- specification  F (g1 & g2)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -- Loop starts here
  -> State: 3.1 <-
    state = s4
    g2 = FALSE
    g1 = FALSE
  -> State: 3.2 <-
-- specification  F ((!g1 & g2) ->  X g1)  is true
```

Gambar 4.11: Hasil Verifikasi dengan *Initial* State $\hat{S}_4$

1. Spesifikasi $\bigcirc(\Box g_1)$ yang berarti bahwa pada transaksi berikutnya Agen selalu mendapatkan keuntungan, spesifikasi tidak dipenuhi dengan *counterexample* yaitu terjadinya *looping* di state $S_4 - S_4 - S_4 - ...$ sehingga terdapat path tak berhingga dan state berikutnya tidak pernah ditemukan $(\Box g_1)$ berarti g_1 tidak pernah benar. Hal ini karena pada state S_4 yang berarti $\hat{S}_4$ memiliki label $\emptyset$ menyatakan bahwa pada state tersebut $\{x_1 \leq 250, x_2 \leq 0\}$ sedangkan yang diinginkan adalah state berikutnya selalu $\{x|x_1 > 250\}$, maka spesifikasi tidak dipenuhi.

2. Spesifikasi $\Diamond(\Box g_2)$ yang berarti bahwa pada suatu waktu barang A selalu ada dalam gudang, spesifikasi tidak dipenuhi dengan *counterexample* yaitu terjadinya *looping* di state $S_4 - S_4 - S_4 - ...$ sehingga terdapat path tak berhingga dan setiap state nya tidak pernah benar. Hal ini karena pada state

S_4 yang berarti $\hat{S}_4$ memiliki label $\emptyset$ menyatakan bahwa pada state tersebut $\{x_1 \leq 250, x_2 \leq 0\}$ sedangkan yang diinginkan adalah akhirnya state selalu $\{x | x_2 > 0\}$, maka spesifikasi tidak dipenuhi.

3. Spesifikasi $\diamondsuit(g_1 \wedge g_2)$ yaitu menyatakan suatu saat $(g_1 \wedge g_2)$ selalu benar yang berarti bahwa pada suatu waktu akhirnya Agen memperoleh keuntungan dan tersedianya barang A dalam gudang , spesifikasi tidak dipenuhi dengan *counterexample* yaitu terjadinya *looping* di state $S_4 - S_4 - S_4 - ...$ sehingga terdapat path tak berhingga dan setiap state nya tidak pernah benar. Hal ini karena pada state S_4 yang berarti $\hat{S}_4$ memiliki label $\emptyset$ menyatakan bahwa pada state tersebut $\{x_1 \leq 250, x_2 \leq 0\}$ sedangkan yang diinginkan adalah akhirnya state $\{x_1 > 250, x_2 > 0\}$, maka spesifikasi tidak dipenuhi.

4. Spesifikasi $\diamondsuit((\leftharpoondown g_1 \wedge g_2) \rightarrow \bigcirc g_1)$ yang berarti pada saat Agen tidak mendapat keuntungan tetapi memiliki barang A dalam gudang, maka akhirnya suatu saat Agen dapat memperoleh keuntungan, spesifikasi terpenuhi.

Jadi dengan *initial* state $\hat{S}_4$, spesifikasi yang dipenuhi adalah spesifikasi $\diamondsuit((\leftharpoondown g_1 \wedge g_2) \rightarrow \bigcirc g_1)$.

Sistem transisi abstrak ada yang memenuhi spesifikasi dan ada yang tidak memenuhi spesifikasi. Meskipun sistem transisi abstrak memiliki hubungan dengan sistem transisi kongkrit yang ditunjukkan dengan adanya transisi, tetapi apabila terdapat spesifikasi yang tidak dipenuhi, maka kita tidak dapat mengatakan bahwa sistem transisi kongkrit juga tidak memenuhi. Sebagai contoh untuk spesifikasi $\bigcirc(\Box g_1)$. Pada sistem transisi kongkrit misalkan terdapat $\{x_1 > 250, x_2 > 0\}$ yang berarti bahwa uang yang dimiliki lebih dari 250 dan terdapat barang A dalam gudang (diasumsikan ada sebuah barang A), transisi jual A dapat di*fire* bergantian dengan transisi Beli A. Pada transaksi berikutnya jumlah uang selalu bertambah karena harga jual lebih besar dari harga beli. Lain halnya dengan apabila sistem transisi abstrak memenuhi spesifikasi, maka sistem transisi kongkrit juga memenuhi spesifikasi tersebut.

DAFTAR PUSTAKA

[1] Adzkiya, D., (2014), *Finite Abstraction of Max-Plus-Linear Systems*, Disertasi, Technische Universiteit Delft, Deflt.

[2] Baier, C., dan Katoen, J. P., (2007), *Principles of Model Checking*, The MIT Press, Cambrigdge.

[3] Cassandras, C. G., dan Lafortune, S., (2008), *Introduction to Discrete Event Systems Second Edition*, New York: Springer.

[4] Murata, T., (1989), Petri Nets: Properties, Analysis and Applications, *Proceding of The IEEE*, hal. 541-580.

[5] Pommereau, F., Devillers, R., dan Klaudel, H., (2009), Efficient Reachability Graph Representation of Petri Nets With Unbounded Counters, *Electronic Notes in Theoretical Computer Science*, Vol. 239, hal. 119-129.

[6] Subiono, (2015), *Aljabar Min-Max Plus dan Terapannya*, Institut Teknologi Sepuluh Nopember, Surabaya.

[7] Szpyrka, M., Biernacka, A., dan Biernacki, J., (2014), Methods of Translation of Petri Nets to NuSMV Language, Ed: Zeugman, L., *CEUR Workshop Proceedings*, Germany, Vol. 1269, hal. 245-256.

BIODATA PENULIS

Penulis yang memiliki nama lengkap Ruvita Iffahtur Pertiwi lahir di Malang, 1 Januari 1992. Nama panggilannya adalah Vita. Penulis telah menempuh pendidikan formal mulai dari SD Negeri 1 Mergosono, SMP Negeri 9 Malang, dan SMA Negeri 5 Malang. Setelah lulus SMA penulis melanjutkan S1 di Jurusan Matematika Universitas Brawijaya Malang sebagai mahasiswa angkatan 2010. Selama kuliah S1 penulis aktif dalam berbagai macam kegiatan seperti menjadi anggota organisasi dan mengikuti kepanitiaan di dalam kampus. Kuliah di Jurusan Matematika penulis tertarik dengan Bidang Minat Aljabar, sehingga Skripsi penulis mengambil topik Aljabar dengan judul "GRUP *NON*-ABELIAN TRIVIAL *INTERSECTION"*. Penulis lulus sarjana delapan semester dengan mendapat gelar Sarjana Sains. Penulis melanjutkan studi S2 di Jurusan Matematika Institut Teknologi Sepuluh Nopember Surabaya pada tahun 2014 semester genap. Pada perkuliahan di S2 penulis tertarik dengan mata kuliah Topik Komputasi dan Aljabar Min-Max Plus. Hal ini menjadi latar belakang penulis untuk mengambil topik Tesis yang berkaitan dengan keduanya, sehingga terbentuklah Tesis ini yang berjudul "VERIFIKASI FORMAL PETRI NET DENGAN *COUNTER* PADA SISTEM INVENTORI". Penulis menyelesaikan pendidikan S2 selama tiga semester dengan mendapat gelar Magister Sains. Jika ingin mengenal lebih dekat, membentuk jaringan, atau membutuhkan informasi yang berhubungan dengan tesis ini, penulis dapat dihubungi melalui email ruvitapertiwi@gmail.com.