Verifikasi Formal Petri Net *Counter* Pada Sistem Inventori

Ruvita Iffahtur Pertiwi, Dieky Adzkiya, Subiono
Jurusan Matematika FMIPA ITS Surabaya
Kampus ITS Sukolilo Surabaya, 60111
Ruvita14@mhs.matematika.its.ac.id

Abstrak — Verifikasi formal merupakan cara untuk memeriksa apakah sistem memenuhi spesifikasi atau tidak. Petri net dengan *counter* (PNZ) merupakan salah satu alat untuk memodelkan sistem *event* diskrit yang dilengkapi representasi data dengan variabel-variabelnya adalah bilangan bulat. Pada penelitian ini dikonstruksi PNZ dari suatu sistem inventori. Pada sistem ini Agen melakukan pembelian barang dan penjualan barang dengan terdapat dua barang yaitu barang A dan barang B. Variabel-variabel dan konstanta pada sistem inventori ini merupakan bilangan bulat tak negatif. Model PNZ sistem inventori pada penelitian ini memiliki state tak berhingga, sehingga tidak bisa diverifikasi dengan *software*. Agar verifikasi bisa dilakukan dengan *software*, digunakan metode abstraksi berhingga. Abstraksi berhingga membuat sistem dengan state tak berhingga (pada sistem transisi kongkrit) menjadi berhingga (pada sistem transisi abstrak). Verifikasi formal membutuhkan spesifikasi formal, dimana spesifikasi formal yang sesuai untuk permasalahan pada penelitian ini adalah *Linear Temporal Logic* (LTL). Jika sistem transisi abstrak memenuhi spesifikasi, maka sistem transisi kongkrit juga memenuhi spesifikasi. Beberapa spesifikasi yang diberikan pada penelitian ini antara lain yang pertama yaitu ”pada transaksi berikutnya Agen selalu mendapat keuntungan”, spesifikasi yang kedua, ”pada suatu waktu barang A selalu ada dalam gudang”, spesifikasi ketiga ”pada suatu waktu, akhirnya Agen memperoleh keuntungan dan tersedianya barang A dalam gudang”, dan spesifikasi keempat ”pada saat Agen tidak mendapat keuntungan tetapi memiliki barang A dalam gudang, maka akhirnya Agen dapat memperoleh keuntungan”. Pada penelitian ini *software* yang digunakan untuk verifikasi adalah NuSMV.

Kata Kunci: Petri net dengan *counter*, sistem *event* diskrit, sistem transisi, verifikasi formal, abstraksi berhingga, LTL, NuSMV.

I. PENDAHULUAN

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 menggunakan *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, verifikasi menggunakan NuSMV untuk memperoleh hasil verifikasi dari sistem apakah telah memenuhi spesifikasi.

II. TINJAUAN PUSTAKA

A. Petri Net

Petri net merupakan salah satu alat untuk memodelkan sistem *event* diskrit.

- Definisi dan Notasi dalam Petri Net

Secara matematis Petri Net dapat dituliskan sebagai 5-*tuple* (P, T, A, ω, x) dengan:

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

Dalam penelitian ini digunakan lingkaran untuk merepresentasikan sebuah *place* dan persegi panjang untuk merepresentasikan sebuah transisi [6].

- Petri Net Bertanda (*Marking*)

Dalam notasi matematika penanda (*marking*) x pada Petri net bertanda (P, T, A, W) adalah fungsi $x : P \to N = \{0,1,2, \dots\}$. Penanda dinyatakan dengan vektor kolom yang elemen-elemennya merupakan bilangan bulat tak negatif (banyaknya token dalam suatu *place*), yaitu $x = [x(p_1); x(p_2), \dots, x(p_n)]^T$. Suatu transisi $t_j \in T$ dalam sebuah Petri net dikatakan kondisinya terpenuhi (*enabled*) jika $x(p_i) \geq \omega(p_i, t_j)$ untuk setiap $p_i \in I(t_j)$.

- 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*). Suatu fungsi perubahan keadaan pada sebuah Petri net (P, T, A, ω, x), yaitu $f: \mathbb{N}^n \times T \to \mathbb{N}^n$ terdefinisi untuk transisi $t_j \in T$ jika dan hanya jika $x(p_i)\omega(p_i, t_j)$ untuk semua $p_i \in I(t_j)$.

B. 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 $\mathbb{V} = \{x_0, \dots, x_{n-1}\}$ adalah himpunan berhingga dari variabel-variabel sedemikian sehingga $\mathbb{V} \cap \mathbb{D} = \emptyset$. Dipilih asumsi untuk memilih variabel yang dituliskan $X = \{x_0, \dots, x_{n-1}\} \in \mathbb{V}^n$ 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 1. Contoh Petri Net dengan *Counter*

Contoh 1. Suatu contoh Petri net dengan *counter* diberikan pada **Gambar 1**, dengan asumsi $X = [x]$, data awal $D_0 = \{[0]\}$ dan setiap transisi dilabeli dengan transformasi linier. Parameter ω adalah bilangan bulat genap tidak negatif. Transformasi linier dari masing-masing transisi adalah

$$L(t_1) = \left(x < \frac{\omega}{2}, x := x + 1\right),$$
$$L(t_2) = (x > 0, x := x - 1),$$
$$L(t_3) = (x < \omega, x := x + 1),$$
$$L(t_4) = (x < \omega, x := x + 1),$$
$$L(t_5) = (x > 0, x := x - 1).$$

C. Verifikasi Formal

Pada penelitian ini akan dilakukan verifikasi formal, dimana verifikasi formal merupakan pengecekan apakah sebuah sistem memenuhi kebutuhan atau tidak. Salah satu teknik yang digunakan adalah model *checking*.

Langkah pertama yaitu memodelkan sistem. Model standart yang digunakan adalah sistem transisi. Sistem transisi berguna untuk mendiskripsikan bagaimana dinamika dari sistem secara matematis sistem transisi adalah 6-$tuple$ $(S, Act, \to, I, AP, Lb)$ dengan $S = s_1, s_2, \dots$ adalah himpunan dari state-state, Act adalah himpunan dari *action*, $\to S \times Act \times S$ adalah relasi transisi, I adalah himpunan dari Initial state, AP adalah himpunan *atomic proposition*, dan $Lb \to 2^{AP}$ adalah fungsi labeling, dengan 2^{AP} merupakan himpunan kuasa dari AP [2].

D. Linear Temporal Logic (LTL)

Linear Temporal Logic (LTL) adalah salah satu formalisasi *logic* yang tepat 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. Bagaimana menunjukkan LTL dapat digunakan untuk spesifikasi properti sistem didefinisikan sintaksis dan semantik dari LTL sebagai berikut.

- **Sintaksis LTL**

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

Definisi Sintaksis LTL [2]

Formula LTL atas himpunan dari atomik preposisi terbentuk mengikuti aturan berikut.

$$\phi ::= true \mid a \mid \phi 1 \wedge \phi 2 \mid \phi \mid \phi 1 \cup \phi 2, a \in AP. \quad (2.1)$$

Formula LTL juga dibangun dari operator dasar dan operator temporal yang telah dijelaskan sebelumnya. Sintaksis LTL akan direpresentasikan dalam tabel sebagai berikut.

Tabel 2.1 Sintaksis

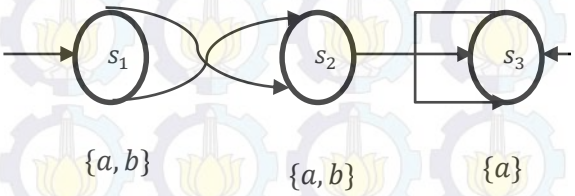
Textual	Simbolik	Keterangan
X	◯	selanjutnya
$X\varphi$	◯φ	formula (φ) selanjutnya
	∪	sampai
	$\varphi \cup \psi$	φ sampai ψ benar
	¬	negasi/ ingkaran
	φ_i	formula ke-i
	∧	dan
	∨	atau
	→	sampai
	↔	ekivalen
G	□	Selalu
$G\ \varphi$	□φ	*always* formula φ
F	◇	*eventually* (akhirnya)
$F\ \varphi$	◇ φ	*eventually* φ
$GF\varphi$	□ ◇ φ	*infinitly often* φ
$FG\varphi$	◇ □ φ	*eventually forever* φ

Contoh 2. Misalkan pada *Trafik Light*, , $\square\left(\boldsymbol{red} \rightarrow \left(\diamond \boldsymbol{green} \wedge (\neg \boldsymbol{green} \cup \boldsymbol{yellow})\right)\right)$ yang artinya sekali lampu berwarna merah, lampu akan selalu hijau akhirnya setelah lampu kuning beberapa kali.

- **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.

Atomic prop a artinya suku atau state yang pertama harus memenuhi a. Formula a menyatakan step selanjutnya atau pada suku ke 2 harus memenuhi a. Formula a ∪ b artinya a harus berlaku dan b tidak berlaku sampai b berlaku, dan seterusnya [1].



Gambar 2. Contoh Sistem Transisi Sederhana

Contoh 3. Diberikan sebuah sistem transisi dengan AP = {a, b} pada **Gambar 2** verifikasi formal sebagai berikut.

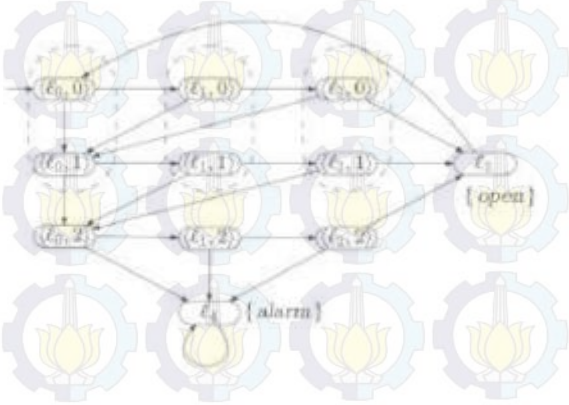
1. $TS \vDash \square\, a$ benar, karena semua state dilabeli dengan a
2. $s_1 \vDash \bigcirc (a \wedge b)$ benar, karena $s_1 \vDash a \wedge b$, , karena $s_1 \vDash a$ dan $s_1 \vDash b$.

E. Abstraksi Berhingga

Abstraksi merupakan konsep yang digunakan untuk menganalisa atau memverifikasi sistem transisi dengan state yang besar atau tak berhingga. Abstraksi membuat state tak berhingga menjadi berhingga atau memperkecil jumlah state. 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]. Sifat-sifat dari abstraksi salah satunya adalah jika abstraksi tidak terpenuhi, maka tidak dapat disimpulkan sistem transisi asli nya juga tidak terpenuhi.

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].

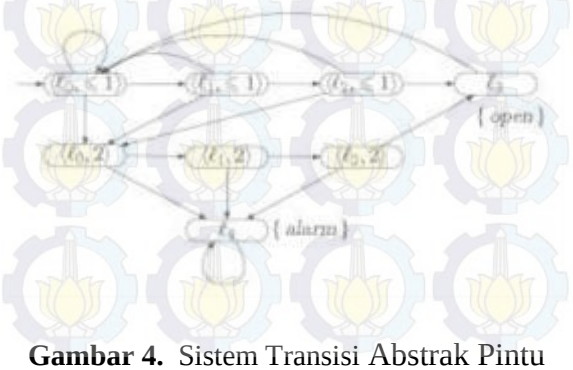


Gambar 3. Contoh Sistem Transisi Pintu Otomatis

Contoh 4. Diberikan sistem transisi pintu otomatis diperlihatkan pada **Gambar 3.** 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 $\ell_{\{1,2,3\}} \in \{0, \ldots, 9\}$ dan ℓ_i untuk $i = 0, 1, 2$. Digit i pertama menyatakan kode yang dimasukkan benar, komponen kedua menyatakan berapa kali melakukan kesalahan. Selanjutnya dilakukan abstraksi berdasarkan fungsi abstrak, sehingga sistem transisi pintu otomatis menjadi



Gambar 4. Sistem Transisi Abstrak Pintu Otomatis

F. 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 [7].

III. PEMBAHASAN

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 (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 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.

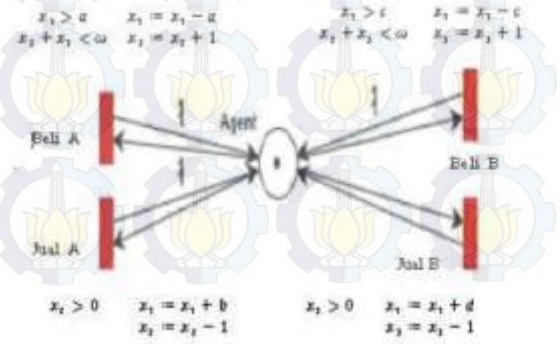
Contoh 5. Sistem inventori ini berupa pembelian barang, penyimpanan barang pada suatu gudang, dan penjualan kembali barang yang disimpan. Barang yang diperdagangkan hanya dua barang yaitu Produk A dan Produk B dengan masing-masing satu satuan volume. Sehingga memiliki empat sistem transisi dan keempat sistem transisi *enable*. Tidak terdapat biaya pemesanan dan penyimpanan. Produk A dan Produk B disimpan dalam satu gudang dengan kapasitas tertentu.

Variabel-variabel yang digunakan PNZ pada Gambar 5 yaitu x_1 adalah jumlah uang, x_2 adalah jumlah Produk A, x_3 adalah jumlah Produk B. Beberapa konstanta diberikan yaitu a adalah harga beli Produk A, b adalah harga jual Produk A, c adalah harga beli Produk B, d adalah harga jual Produk B, dan ω merupakan kapasitas gudang. Transisi Beli Produk A menyatakan Agen membeli Produk A, dapat difire apabila memenuhi kondisi uang yang dimiliki melebihi harga beli Produk A dan jumlah barang yang dimiliki kurang dari kapasitas gudang, sehingga berpengaruh pada berkurangnya uang yang dimiliki sedangkan jumlah Produk A bertambah. Begitu juga dengan transisi Beli Produk B, seperti Beli Produk A. Transisi Jual Produk A menyatakan Agen menjual kembali Produk A, dapat difire apabila kondisi Agen memiliki Produk A dalam gudang terpenuhi, sehingga berpengaruh pada bertambahnya uang yang dimiliki sedangkan jumlah barang berkurang.

A. Sistem Transisi

Sistem transisi (TS) bermula dari *Initial* state $(s_0 \in I)$ dan dinamikanya diberikan oleh relasi transisi. Definisi sistem transisi telah diberikan pada bagian sebelumnya, berikut diberikan definisi transisi yang disesuaikan dengan PNZ. memenuhi kondisi

1. $FD \leq Q$ yang menyatakan kondisi linier dari PNZ untuk t_j .
2. $Mi \geq w(p_i, t_j)$ untuk setiap $p_i i \in I(t_j)$.
3. $M' = M + Ae_j$ dengan e_j merupakan matriks kolom ke$-j$ dari matriks identitas



Gambar 5. PNZ Sistem Inventori

berorder $|P|$ dan matriks *incidence* $A = Af - Ab$,

4. $D' = KD + B$ yang menyatakan *update* linier dari PNZ untuk transisi t_j.

Artinya, terdapat transisi pada PNZ yang difire sehingga mengakibatkan marking dan data berubah dari (M, D) ke (M', D').

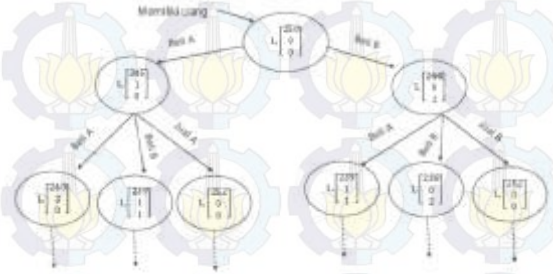
• himpunan dari *initial* state I adalah $\{(M0, D0)\}$.

• himpunan AP diasumsikan berhingga,

• dan diasumsikan

$$L^{-1}b\,(a) = \{(M, D)|FiD \leq Qi\}.$$

Contoh 6. Akan diberikan sistem transisi yang dibangun oleh PNZ sistem inventori yang ada pada Contoh 5. 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 $d = 8$.



Gambar 6. Contoh Sisem Transisi yang Dibangun dari PNZ

Sistem transisi pada **Gambar 6** memiliki *Initial* state (M_0, D_0) dengan $M_0 = 1$ dan $D = [250, 0, 0]$. *Initial* state menjelaskan tentang seorang Agen memiliki uang sejumlah 250. State yang dapat terjadi yaitu Beli Produk A atau Beli Produk B, transisi dari state $(1, [250, 0, 0])$ ke state $(1, [245, 1, 0])$ atau state $(1, [244, 0, 1])$. Misalkan state selanjutnya adalah Beli A maka statenya menjadi $(1, [250, 0, 0])$, 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 Produk A atau 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 6**.

B. Abstraksi Berhingga

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 banyaknya anggota dari himpunan AP.

- **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 dengan cara mempartisi AP. Langkah-langkahnya yaitu:

• menentukan AP dari PNZ,

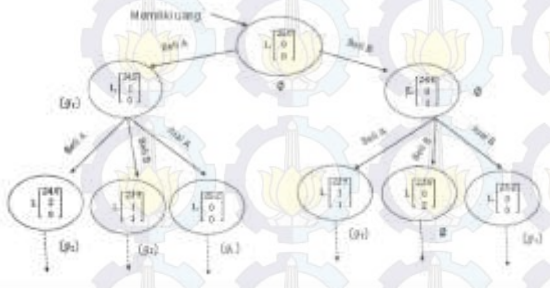
• melabeli setiap state PNZ dengan AP yang sesuai,

• mempartisi state sesuai label yang sama,

• diperoleh state abstrak.

Contoh 7. Diberikan AP dari **Contoh 6** 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 6** yaitu$(1, [250, 0, 0]) = s_1$,...dan seterusnya, sehingga label (L_b) dari state pada sistem transisi sebagai berikut:$Lb(s_1) = \emptyset$, ...

dan seterusnya.



Gambar 7. Pelabelan Sisem Transisi yang Dibangun dari PNZ

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

- $g_1 \wedge g_2 = \overline{S_1} = \{(1,[x_1,x_2,x_3])|x_1 > 250, x_2 > 0\}$
- $g_1 \wedge \leftarrow g_2 = \overline{\overline{S_2}} = \{(1,[x_1,x_2,x_3])|x_1 > 250, x_2 \leq 0\}$
- $\leftarrow g_1 \wedge g_2 = \overline{\overline{S_3}} = \{(1,[x_1,x_2,x_3])|x_1 \leq 250, x_2 > 0\}$
- $\leftarrow g_1 \wedge \leftarrow g_2 = \overline{\overline{S_4}} = \{(1,[x_1,x_2,x_3])|x_1 \leq 250, x_2 \leq 0\}$

Selanjutnya didefinikan fungsi abstrak $f : S \rightarrow \hat{S}$ sedemikian sehingga

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

Berdasarkan partisi diperoleh empat state abstrak yaitu $\hat{S}_1$, $\hat{S}_2$, $\hat{S}_3$, dan $\hat{S}_4$. 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 $s_1 \rightarrow \hat{S}_4$,
- dan seterusnya.

Semua state dapat dipetakan ke state abstrak. 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.

- **Menentukan Transisi pada Sistem Transisi Abstrak**

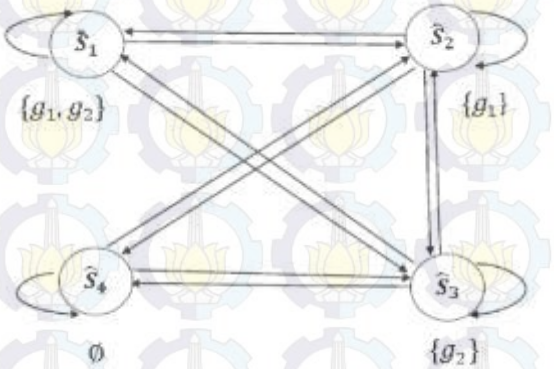
Apabila pada sistem transisi terdapat transisi dari state s ke state s' yang dinotasikan dengan $s \rightarrow s'$, maka juga terdapat transisi yang besesuaian pada state-state abstrak $\hat{s} \rightarrow_f \hat{s'}$. Untuk dapat mengetahui transisi-transisi yang mungkin, digunakan fungsi abstraksi f seperti yang ada pada sebelumnya sedemikian sehingga $f^{-1}(\hat{s}) = \{s : f(s) = \hat{s}$.

Contoh 8. Berdasarkan **Contoh 6** dan **7**, dapat dibuat sistem transisi abstrak. Diketahui terdapat empat state abstrak $\widehat{S_1}$, $\widehat{S_2}$, $\widehat{S_3}$, dan $\widehat{S_4}$, akan ditunjukkan transisi-transisi pada state abstrak sebagai berikut.

$f^{-1}(\widehat{S_1}) = \widehat{S_1} = \{(1,[x_1,x_2,x_3])|x_1 > 250, x_2 > 0\}$
$f^{-1}(\widehat{S_2}) = \widehat{S_2} = \{(1,[x_1,x_2,x_3])|x_1 > 250, x_2 \leq 0\}$
$f^{-1}(\widehat{S_3}) = \widehat{S_3} = \{(1,[x_1,x_2,x_3])|x_1 \leq 250, x_2 > 0\}$
$f^{-1}(\widehat{S_4}) = \widehat{S_4} = \{(1,[x_1,x_2,x_3])|x_1 \leq 250, x_2 \leq 0\}$.

1. Transisi-transisi yang mungkin dari $\widehat{S_1}$ adalah $\widehat{S_1} \rightarrow \widehat{S_1}, \widehat{S_1} \rightarrow \widehat{S_2}$, dan $\widehat{S_1} \rightarrow \widehat{S_3}$.
2. Transisi-transisi yang mungkin dari $\widehat{S_2}$ adalah $\widehat{S_2} \rightarrow \widehat{S_1}, \widehat{S_2} \rightarrow \widehat{S_2}, \widehat{S_2} \rightarrow \widehat{S_3}$, dan $\widehat{S_2} \rightarrow \widehat{S_4}$.
3. Transisi-transisi yang mungkin dari $\widehat{S_3}$ adalah $\widehat{S_3} \rightarrow \widehat{S_1}, \widehat{S_3} \rightarrow \widehat{S_2}, \widehat{S_3} \rightarrow \widehat{S_3}$, dan $\widehat{S_3} \rightarrow \widehat{S_4}$.
2. Transisi-transisi yang mungkin dari $\widehat{S_4}$ adalah $\widehat{S_4} \rightarrow \widehat{S_2}, \widehat{S_4} \rightarrow \widehat{S_3}$, dan $\widehat{S_4} \rightarrow \widehat{S_4}$.

Berdasarkan kemungkinan transisi yang terjadi pada state abstrak, diperoleh sistem transisi abstrak



Gambar 8. Contoh Sistem Transisi Abstrak

C. Spesifikasi

Tujuan dari verifikasi suatu model sistem adalah untuk mengetahui apakah model yang dimiliki telah sesuai dan memenuhi spesifikasi yang diinginkan. Spesifikasi bernilai benar (*true*) atau salah (*false*). Spesifikasi dibuat berdasarkan keinginan agar model dapat merepresentasikan sistem sesuai dengan

kebutuhan, sehingga dapat meminimalisir kesalahan sistem.

Contoh 9. Diberikan spesifikasi untuk verifikasi formal PNZ berdasarkan contoh-contoh sebelumnya. Diketahui suatu sistem pada **Contoh 7** dengan $AP = \{g_1, g_2\}$ 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, maka beberapa spesifikasi yang dapat digunakan adalah:

• Pada transaksi berikutnya Agen selalu mendapatkan keuntungan, formulanya nya adalah $\bigcirc \{x|x_1 > 250\}$ atau $\bigcirc (\square g_1)$.

• Pada suatu waktu barang A selalu ada dalam gudang, formulanya adalah $\Diamond(\square\{x|x_2 > 0\})$ atau $\Diamond (\square\, g_2)$.

• Pada suatu waktu yang sama Agen memperoleh keuntungan dan tersedianya barang A dalam gudang , formulanya adalah $\Diamond (\{x|x_1 > 250\} \wedge \{x|x_2 > 0\})$ atau $\Diamond \varphi(g_1 \wedge g_2)$.

• Pada saat Agen tidak mendapat keuntungan tetapi memiliki barang A dalam gudang, maka suatu saat Agen dapat memperoleh keuntungan, formulanya adalah $\Diamond ((\{x|x_1 \leq 250\} \wedge \{x|x_2 > 0\}) \rightarrow \bigcirc\{x|x_1 > 250\})$ atau $\Diamond ((\leftarrow g_1 \wedge g_2) \rightarrow \bigcirc g_1)$.

D. Implementasi dengan NuSMV

Verifikasi formal dapat dilakukan menggunakan *software* model *checker*, dengan mendefiniskikannya terlebih dahulu dalam bahasa NuSMV. Setelah didefiniskan Pada NuSMV, spesifikasi dapat dicek dan dievaluasi. Ketika spesifikasi salah atau tidak terpenuhi, NuSMV memberikan *counterexample*.

Contoh 10. Didefinisikan sistem transisi dalam bahasa NuSMV pada **Gambar 9.** Berdasarkan sistem transisi abstrak pada **Gambar 8** dan spesifikasi pada **Contoh 9**

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

Gambar 9. Contoh Bahasa NuSMV

Apabila *initial* state adalah $\widehat{S_1}$ yang berlabel $L_b = \{g_1, g_2\}$ berarti bahwa awalnya Agen telah memiliki uang lebih dari 250 dengan $\{x|x_1 > 250\}$ dan terdapat barang A dalam gudang dengan $\{x|x_2 > 0\}$. Akan ditunjukkan apakah sistem memenuhi atau tidak spesifikasi-spesifikasi pada **Contoh 9** sehingga diperoleh hasil verifikasi formal yaitu

```
-- 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 = s2
    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 10. Hasil Verifikasi pada NuSMV

• Spesifikasi $\bigcirc (\square g_1)$ yang berarti bahwa pada transaksi berikutnya Agen selalu mendapatkan keuntungan,spesifikasi tidak dipenuhi dengan contoh *counterexample*

yaitu pada state 1.2 state S_3 dituliskan $g_1 = false$ karena pada S_3 yang berarti $\widehat{S_3}$ memliki label $\{g_2\}$ berarti bahwa pada state tersebut $\{x_1 \leq 250, x_2 > 0\}$ sedangkan yang selalu $\{x|x_1 > 250\}$, maka spesifikasi tidak dipenuhi.

• Spesifikasi $\Diamond(\Box g_2)$ yang berarti bahwa pada suatu waktu barang A selalu ada dalam gudang, spesifikasi tidak dipenuhi dengan contoh *counterexample* yaitu pada state 2.2 state S_2 dituliskan $g_2 = false$ karena pada S_2 yang berarti $\widehat{S_2}$ memliki label $\{g_1\}$ berarti 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.

• Spesifikasi $\Diamond \varphi(g_1 \wedge g_2)$ yang berarti bahwa pada suatu waktu yang sama akhirnya Agen memperoleh keuntungan dan tersedianya barang A dalam gudang , terpenuhi. Sebagai contoh state S_1 ke S_1.

• Spesifikasi $\Diamond((\leftarrow g_1 \wedge g_2) \rightarrow \bigcirc g_1)$ yang berarti pada saat Agen tidak mendapat keuntungan tetapi memiliki barang A dalam gudang, maka akhirnya suatu saat Agen dapat memperoleh keuntungan, terpenuhi. Sebagai contoh yang memenuhi $(\leftarrow g_1 \wedge g_2)$ adalah state S_3, terdapat transisi $S_3 \rightarrow S_1$ dan $S_3 \rightarrow S_2$ sedemikian sehingga memenuhi dari S_3 yang berarti $\{x_1 \leq 250, x_2 > 0\}$ berikutnya dapat menjadi $\{x_1 > 250, x_2 > 0\}$ atau menjadi $\{x_1 > 250\}$).

Jadi dengan *initial* state $\widehat{S_1}$, sistem transisi abstrak ada yang memenuhi spesifikasi dan ada yang tidak memenuhi spesifikasi. Meskipun sistem transisi abstrak memiliki hubungan dengan sistem transisi asli yang ditunjukkan dengan adanya transisi, tetapi apabila terdapat spesifikasi yang tidak dipenuhi, maka kita tidak dapat mengatakan bahwa sistem transisi asli juga tidak memenuhi. Sebagai contoh untuk spesifikasi $\bigcirc(\Box g_1)$. Pada sistem transisi asli misalkan terdapat $\{x_1 > 250, x_2 > 0\}$ yang berarti bahwa uang yang dimiliki lebih dari 250 dan terdapat barang A dalam gudang (diasumsikan jumlah barang A sangat banyak), transisi jual A dapat di*fire* terus menerus (apabila kondisi linier selalu dipenuhi) sehingga transaksi berikutnya jumlah uang selalu bertambah yang berarti selanjutnya Agen selalu mendapat untung. Lain halnya dengan apabila sistem transisi abstrak memenuhi spesifikasi, maka sistem transisi asli pasti juga memenuhi spesifikasi tersebut.

Verifikasi diujikan dengan semua kemungkinan *initial* state pada state-state transisi abstrak.

4. KESIMPULAN

Berdasarkan pembahasan yang telah dilakukan maka dapat diambil kesimpulan sebagai berikut:

1. Model PNZ sistem inventori pada penelitian ini terdiri dari empat transisi dan satu *place*, dimana masing-masing transisi dilengkapi dengan kondisi linear dan *update linear*.
2. Dengan menerapkan metode abstraksi berhingga, sistem transisi dari PNZ yang memiliki jumlah state tak berhingga dapat dibentuk menjadi sistem transisi abstrak dengan jumlah state berhingga.
3. Berdasarkan implementasi pada NuSMV, diperoleh hasil verifikasi terhadap sistem transisi abstrak. Jika sistem transisi abstrak memenuhi spesifikasi LTL, maka sistem transisi kongkritnya juga memenuhi spesifikasi.

5. 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), *Principle of Model Checking*, The Mit Press, Cambrigde.

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

[4] Murata, T., (1989), *Petri Net: Properties, Analysis adn Applications*, Proceding of The IEEE, hal. 541-580.
[5] Pommereau, F., Devillers, R., dan Klaudel, H., (2009), *Efficient Reacheability Graph Representation of Petri Nets With Unbounded Counters*, Elektronic Notes in Theoritical Computer Science, Vol. 239, hal. 119-129.
[6] Subiono, (2015), *Aljabar Min-Max Plus dan Terapannya*, Bahan Kuliah: Aljabar Min Max Plus, Institut Teknologi Sepuluh Nopember, Surabaya
[7] Szpyrka, M., Biernacka, A., dan Bienarcki, J., (2014), *Methodes Of Translations of Petri Nets to NuSMV Language, Ed: Zeugman, L.*, Ceur Workhop proceedings, Germany, Vol. 1269, hal. 245-256.