TUGAS AKHIR – SM141501

VERIFIKASI FORMAL DESAIN SISTEM TRANSAKSI ATM (*AUTOMATED TELLER MACHINE*) MENGGUNAKAN SPIN (*SIMPLE PROMELA INTERPRETER*)

IKHWAN MOHAMMAD IQBAL
1212100086

Dosen Pembimbing :
Dr. Dieky Adzkiya, S.Si, M.Si
Dr. Imam Mukhlash, S.Si, M.T

JURUSAN MATEMATIKA
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT – SM141501

FORMAL VERIFICATION OF ATM (AUTOMATED TELLER MACHINE) TRANSACTION SYSTEM DESIGN USING SPIN (SIMPLE PROMELA INTERPRETER)

IKHWAN MOHAMMAD IQBAL
1212100086

Supervisors :
Dr. Dieky Adzkiya, S.Si, M.Si
Dr. Imam Mukhlash, S.Si, M.T

DEPARTMENT OF MATHEMATICS
Faculty of Mathematics and Natural Sciences
Sepuluh Nopember Institute of Technology
Surabaya 2016

LEMBAR PENGESAHAN

VERIFIKASI FORMAL DESAIN SISTEM TRANSAKSI ATM (AUTOMATED TELLER MACHINE) MENGGUNAKAN SPIN (SIMPLE PROMELA INTERPRETER)

FORMAL VERIFICATION OF ATM (AUTOMATED TELLER MACHINE) TRANSACTION SYSTEM DESIGN USING SPIN (SIMPLE PROMELA INTERPRETER)

Diajukan Untuk Memenuhi Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Sains
pada
Bidang Studi Ilmu Komputer
Program Studi S-1 Jurusan Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember Surabaya
Oleh :
IKHWAN MOHAMMMAD IQBAL
NRP. 1212100086
Menyetujui,

Dosen Pembimbing II,

Dr. Imam Mukhlash, S.Si, M.T
NIP. 19700831 199403 1 003

Dosen Pembimbing I,

Dr. Dieky Adzkiya,S.Si M.Si
NIP. 19830517 200812 1 003

Mengetahui,
Ketua Jurusan Matematika
FMIPA ITS

Dr. Imam Mukhlash, S.Si, MT
NIP. 19700831 199403 1 003

Surabaya, Juni 2016

VERIFIKASI FORMAL DESAIN SISTEM TRANSAKSI ATM (*AUTOMATED TELLER MACHINE*) MENGGUNAKAN SPIN (*SIMPLE PROMELA INTERPRETER*)

Nama Mahasiswa	:	Ikhwan Mohammad Iqbal
NRP	:	1212100086
Jurusan	:	Matematika FMIPA - ITS
Pembimbing	:	1. Dr. Dieky Adzkiya, S.Si, M.Si
		2. Dr. Imam Mukhlash, S.Si, M.T

ABSTRAK

Verifikasi formal adalah teknik penting untuk memastikan kebenaran sistem yang diselidiki. Namun, tools pada model pemeriksaan harus memperhatikan perkembangan ruang state dimana hal ini akan mengabaikan teknik praktis dalam teknik penerapannya. Pemodelan dan deskripsi dari Automated Teller Machine (ATM) merupakan pemodelan kasus desain yang khusus dalam sistem yang real time dan safety critical. Sebagai sebuah sistem kontrol yang real time, sistem ATM merupakan sistem dengan tingkat kompleksitas yang tinggi. SPIN (Simple Promela Interpreter) adalah sistem verifikasi generik yang mendukung desain dan proses sistem verifikasi yang asynchronous. Model Pemeriksaan ini, mampu menerima spesifikasi desain yang ditulis dalam bahasa verifikasi PROMELA (Process Meta Language), dan hal tersebut mampu menerima klaim kebenaran atau claim correctness yang ditentukan dalam sintaks dalam LTL (Linear Temporal Logic) sederhana. Dalam penelitian ini akan dilakukan verifikasi terhadap model sistem ATM dengan SPIN model checker

***Kata kunci** :Verifikasi Formal, SPIN, PROMELA, ATM*

FORMAL VERIFICATION OF ATM (AUTOMATED TELLER MACHINE) TRANSACTION SYSTEM DESIGN USING SPIN (SIMPLE PROMELA INTERPRETER)

Name	:	Ikhwan Mohammad Iqbal
NRP	:	1212100086
Department	:	Mathematics FMIPA - ITS
Supervisors	:	1. Dr. Dieky Adzkiya, S.Si, M.Si
		2. Dr. Imam Mukhlash, S.Si, M.T

ABSTRACT

Formal verification is an important technique for ensuring the correctness of investigated systems. However, the model checking tools subject to the state-space explosion problem, which is an ignored hurdle to the practical application of the technique. The modeling and description of an Automated Teller Machine (ATM) are a typical design case in safety-critical and real-time systems as a real-time control system, the ATM system is characterized by its high degree of complexity. SPIN(Simple Promela Interpreter) is a generic verification system that supports the design and verification of asynchronous process systems. This model checker accepts design specifications written in the verification language PROMELA (a Process Meta Language), and it accepts correctness claims specified in the syntax of standard Linear Temporal Logic (LTL). In this case, we verifies the ATM model system using SPIN model checker.

***Keywords** : Formal Verification, SPIN, PROMELA, ATM*

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan limpahan rahmat, petunjuk serta hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul

"VERIFIKASI FORMAL DESAIN SISTEM TRANSAKSI ATM (AUTOMATED TELLER MACHINE) MENGGUNAKAN SPIN (SIMPLE PROMELA INTERPRETER)"

sebagai salah satu syarat kelulusan Program Sarjana Jurusan Matematika FMIPA Institut Teknologi Sepuluh Nopember (ITS) Surabaya.

Sholawat serta salam senantiasa penulis curahkan kepada junjungan Nabi besar Muhammad SAW, beserta para keluarga dan sahabatnya. Tugas akhir ini dapat terselesaikan dengan baik berkat bantuan dan dukungan dari berbagai pihak. Oleh karena itu, penulis menyampaikan ucapan terima kasih dan penghargaan kepada:

1. Bapak Dr. Imam Mukhlash, S.Si, MT selaku Ketua Jurusan Matematika ITS serta selaku dosen pembimbing.
2. Bapak Dr. Dieky Adzkiya, S.Si, M.Si selaku dosen pembimbing atas segala bimbingan dan motivasinya kepada penulis dalam mengerjakan tugas akhir ini sehingga dapat terselesaikan dengan baik.
3. Bapak Dr. Mahmud Yunus, M.Si, Bapak Drs. Sadjidon, M.Si dan Bapak Kistoshil Fahim, S.Si, M.Si selaku dosen penguji atas semua saran yang telah diberikan demi perbaikan tugas akhir ini.
4. Bapak Dr. Chairul Imron, MI.Komp. selaku KAPRODI Sarjana Matematika ITS dan Mas Ali yang selalu memberikan informasi mengenai tugas akhir.
5. Bapak Dr. Darmaji, S.Si, M.T selaku dosen wali yang telah memberikan arahan akademik selama penulis menempuh pendidikan di Jurusan Matematika FMIPA ITS.

6. Bapak dan Ibu dosen serta para staf Jurusan Matematika ITS yang tidak dapat penulis sebutkan satu persatu.

Penulis berharap semoga tugas akhir ini dapat bermanfaat bagi banyak pihak.

Surabaya, Juni 2016

Penulis

DAFTAR ISI

LEMBAR PENGESAHAN iv
ABSTRAK vi
ABSTRACT viii
KATA PENGANTAR x
DAFTAR ISI xii
DAFTAR GAMBAR xiv
PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan 4
1.5 Manfaat 5
1.6 Sistematika Penulisan 5
TINJAUAN PUSTAKA 7
2.1 Pemodelan Berorientasi Obyek 7
2.2 Sistem Transisi 9
2.3 LTL (Linear Temporal Logic) 10
2.4 PROMELA 11
2.5 SPIN (*Simple Promela Interpreter*) 13
METODOLOGI PENELITIAN 15
ANALISIS DAN PEMBAHASAN 19
4.1 Mode Simulasi dalam SPIN 19
4.1.1 Simulasi Acak 20

4.1.2 Simulasi Terarah 21
4.1.3 Simulasi *Exhaustive Depth First Search* 21
4.1.4 Mode Simulasi yang digunakan 22
4.2 Asumsi Proses Bisnis Sistem ATM 23
4.3 Pemodelan Sistem ATM dalam Diagram Transisi 24
4.4 Pemodelan Sistem ATM dalam PROMELA 33
4.4.1 Definisi *mtype, channel* dan *boolean* 33
4.4.2 Definisi *proctype Consortium* 35
4.4.3 Definisi *proctype Bank* 38
4.4.4 Definisi *proctype User* 41
4.4.5 Definisi *proctype ATM* 45
4.5 Pemeriksaan Properti 53
PENUTUP 67
5.1 Kesimpulan 67
5.2 Saran 67
DAFTAR PUSTAKA 69
BIODATA PENULIS 71

DAFTAR GAMBAR

Gambar 4. 1 Mode Simulasi SPIN 19
Gambar 4. 2 Hasil Mode Simulasi Acak dalam SPIN 20
Gambar 4. 3 Hasil Mode Simulasi Exhaustive Depth First Search dalam SPIN 22
Gambar 4. 4 Diagram Transisi Sistem ATM 27
Gambar 4. 5 Diagram Transisi Sistem ATM Proses Verifikasi Password 27
Gambar 4. 7 Proses ATM dalam PROMELA 29
Gambar 4. 8 mtype Sistem ATM dalam PROMELA 31
Gambar 4. 9 channel Sistem ATM dalam PROMELA 32
Gambar 4.10 Inisialisasi Program PROMELA 33
Gambar 4.11 Definisi mtype, channel dan boolean 35
Gambar 4.12 Definisi proctype Consortium 38
Gambar 4.13 Definisi proctype Bank 40
Gambar 4.14 Definisi proctype User 44
Gambar 4.15 Definisi proctype ATM 52
Gambar 4.16 Hasil Ekesekusi Program 55
Gambar 4.17 Hasil Eksekusi untuk Properti 1 58
Gambar 4.18 Hasil Eksekusi untuk Properti 2 59
Gambar 4.19 proses ReqContinueState ATM 63
Gambar 4.20 Hasil Eksekusi untuk Liveness 1 64
Gambar 4.21 Hasil Eksekusi untuk Safety 1 65

BAB I

PENDAHULUAN

Pada bab ini, dijelaskan mengenai latar belakang penelitian tugas akhir serta rumusan masalah dan batasan masalah berdasarkan latar belakang tersebut. Selain itu, juga dijelaskan tujuan dan manfaat penelitian tugas akhir serta sistematika penulisan tugas akhir.

1.1 Latar Belakang

Bank ritel merupakan sebuah bisnis bank yang penting, yang menawarkan berbagai layanan untuk nasabah, individu atau bisnis kecil menengah, bukan untuk perusahaan besar dan bank lainnya. Layanan biasanya meliputi giro, tabungan, investasi, pinjaman dan hipotek. Saat ini, bank ritel juga menawarkan jasa mereka melalui telepon atau *mobile banking*, internet dan melalui ATM (*Automated Teller Machine)* [1].Fasilitas yang diberikan oleh bank sangatlah beragam sehingga sistem harus berjalan dengan baik. Selain itu, beberapa sektor bisnis lain seperti pasar swalayan menggunakan jasa perbankan, terutama dalam pembayarannya. Maka dari itu, sangat diperlukan verifikasi untuk memastikan kebenaran dari sistem tersebut.

Pemodelan dan deskripsi dari *Automated Teller Machine* (ATM) merupakan pemodelan kasus desain yang khusus dalam sistem yang *real time* dan *safety critical*. Sistem yang *real time* adalah sebuah sistem dimana harus mampu mempertahankan waktu respon atau *wait loop* yang singkat, sedangkan untuk sistem *safety critical* adalah sebuah sistem di mana kegagalan atau kesalahan desain memiliki potensi untuk menyebabkan

permasalahan pada pengguna dengan tingkat urgensi yang sangat tinggi. Sebagai sebuah sistem kontrol yang *real time,* sistem ATM merupakan sistem dengan tingkat kompleksitas yang tinggi, interaksi yang rumit antara perangkat keras dan pengguna dan juga persyaratan yang ketat untuk *domain knowledge*. Hal tersebut membuat sistem ATM menjadi sangat kompleks namun cukup ideal dalam desain sistem perangkat lunak untuk skala besar pada umumnya serta dalam pemodelan sistem *real time* pada khususnya [2].

Terdapat kekurangan dalam sistematika dan dokumentasi detil dari desain dan pemodelan pada prototipe sistem ATM serta kurangnya model formal dari sistem tersebut. Sehingga dalam hal ini terdapat celah adanya kesalahan atau *bug* yang mungkin terjadi selama sistem tersebut berjalan.

Pengujian merupakan langkah yang sangat diperlukan untuk memastikan kebenaran dari suatu sistem. Namun, pengujian tidak pernah bisa benar – benar mengidentifikasi semua kesalahan dalam sebuah sistem. Model Pemeriksaan atau *Model Checking* adalah metode untuk verifikasi formal dengan *state* yang berhingga atau *finite-state* dalam sistem konkuren. Dalam model pemeriksaan, sifat atau *property* mengenai verifikasi sistem biasanya diekspresikan dengan *temporal logic* dan algoritma efisien yang digunakan untuk melintasi model tersebut [3].

SPIN *(Simple Promela Interpreter)* adalah sistem verifikasi generik yang mendukung desain dan proses sistem verifikasi yang *asynchronous.* Model pemeriksaan ini, mampu menerima spesifikasi desain yang ditulis dalam bahasa verifikasi PROMELA *(Process Meta Language)*, dan hal tersebut mampu menerima klaim kebenaran atau *claim correctness* yang ditentukan dalam sintaks dalam LTL *(Linear Temporal Logic)*

sederhana. Bahasa input dari model pemeriksaan SPIN memungkinkan untuk membangun model tingkat tinggi terhadap sistem terdistribusi dari tiga komponen dasar, yaitu proses *asynchronous*, saluran pesan dan obyek data [4].

Model pemeriksaan merupakan pendekatan yang menjanjikan untuk memastikan kebenaran dari sebuah sistem perbankan terutama dalam hal ini adalah dalam sistem ATM. Berdasarkan hasil survey terkait metode verifikasi terhadap model pemeriksaan yang ada menunjukan bahwa metode SPIN cocok digunakan dalam verifikasi model pemeriksaan jenis alur bisnis atau *business flow* [5]. Sehingga, dalam penelitian ini akan digunakan model pemeriksaan untuk memastikan kebenaran dari sistem tersebut dan menggunakan SPIN untuk melakukan verifikasi terhadap model sistem tersebut.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, masalah yang akan dibahas dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana mendefinisikan sistem ATM dalam bentuk sistem transisi dan PROMELA untuk dilakukan proses verifikasi.
2. Apakah model sistem tersebut memenuhi spesifikasi yang diberikan.

1.3 Batasan Masalah

Permasalahan yang dibahas pada Tugas Akhir ini nantinya akan dibatasi ruang lingkup pembahasannya, antara lain:

1. Penulis mendefinisikan model proses bisnis sistem ATM bersumber dari buku *Object Oriented Modeling and Design* [6]

2. Proses pemeriksaan terdiri dari 2 tahap dimana tahap 1 adalah pemeriksaan awal dan tahap 2 adalah pemeriksaan property *Liveness* dan *Safety*. Pada tahap 1 ditentukan untuk beberapa spesifikasi diantaranya adalah :
   a. Jika kas ditiadakan dan pengguna memilih untuk tidak melanjutkan transaksi, maka ATM akhirnya akan mencetak tanda terima dan mengeluarkan kartu.
   b. Jika ATM mencetak tanda terima, maka akan mengeluarkan kartu.

   Sedangkan, pada tahap 2 Pemeriksaan akan ditentukan dengan spesifikasi sebagai berikut :
   a. Jika pengguna memilih menu penarikan uang dan transaksi berhasil, maka ATM akhirnya akan mengeluarkan uang tunai.
   b. Jika ATM mengeluarkan uang tunai, maka akan mencetak tanda terima.
   c. Jika ATM mengeluarkan uang tunai, maka akan mengeluarkan kartu.
   d. Jika pengguna memasukkan pin yang salah, maka ATM tidak akan mengeluarkan uang tunai.
   e. Jika transaksi di ATM gagal, maka ATM tidak akan mengeluarkan uang tunai.
   f. Jika pengguna tidak memilih untuk melakukan transaksi penarikan uang, maka ATM tidak akan mengeluarkan uang tunai.
   g. Jika pengguna memilih untuk melakukan transaksi cek saldo, maka ATM tidak akan mengeluarkan uang tunai

1.4 Tujuan

Tujuan dari Tugas Akhir yang ingin dicapai berdasarkan rumusan masalah yang telah diuraikan adalah sebagai berikut:

1. Mendapatkan definisi sistem ATM dalam bentuk sistem transisi dan PROMELA untuk proses verifikasi.
2. Mengetahui bahwa apakah model sistem tersebut memenuhi spesifikasi yang diberikan.

1.5 Manfaat

Manfaat dalam tugas akhir ini adalah

1. Memberikan hasil desain formal sistem ATM.
2. Memberikan hasil verifikasi pada model sistem, sehingga dapat diketahui kesalahan atau *bug* dalam sistem tersebut.

1.6 Sistematika Penulisan

Sistematika penulisan dalam tugas akhir ini sebagai berikut:

BAB I Pendahuluan, menjelaskan tentang latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika penulisan laporan tugas akhir.

BAB II Tinjauan Pustaka, menjelaskan tentang konsep – konsep yang mendukung analisis dan pembahasan. Diantaranya adalah pemodelan berorientasi obyek, sistem transisi, LTL (*Linear Temporal Logic),* PROMELA dan SPIN (*Simple Promela Interpreter*).

BAB III Metode Penelitian, menjelaskan tentang tahapan – tahapan dalam penelitian tugas akhir.

BAB IV Analisis dan Pembahasan, menjelaskan tentang mode simulasi dalam SPIN, asumsi proses bisnis sistem ATM, pemodelan sistem ATM dalam diagram transisi, pemodelan sistem ATM dalam promela, pemeriksaan properti.

BAB V Penutup, berisi kesimpulan dan saran berdasarkan pembahasan dari seluruh penelitian tugas akhir.

BAB II

TINJAUAN PUSTAKA

Pada bab ini, diuraikan beberapa hal yang mendukung penyelesaian tugas akhir. Beberapa hal tersebut meliputi pemodelan berorientasi obyek, sistem transisi, LTL (*Linear Temporal Logic),* PROMELA dan SPIN (*Simple Promela Interpreter*)

2.1 Pemodelan Berorientasi Obyek

Sebuah model obyek menangkap struktur statis dari sistem dengan mengGambarkan obyek dalam sistem. Model berorientasi obyek lebih mendekati keadaan nyata, dan dilengkapi dengan penyajian grafis dari sistem yang sangat bermanfaat untuk komunikasi dengan user dan pembuatan dokumentasi struktur dari sistem [6].

Dari sisi desain, pemodelan berorientasi obyek menggunakan tiga macam model antara lain :

1. Model Obyek

   Model obyek yaitu mengGambarkan struktur statis dari suatu obyek dalam sistem dan relasinya yang berisi diagram obyek yaitu suatu *graph* dimana *node*-nya adalah kelas yang mempunyai relasi antar kelas.

2. Model Dinamik

   Model dinamik yaitu mengGambarkan aspek dari sistem yang berubah setiap saat yang dipergunakan untuk menyatakan aspek kontrol dari sistem yang berisi *state diagram* yaitu suatu *graph* dimana *node*-nya adalah *state*

dan *arc* adalah tarnsisi antara *state* yang disebabkan oleh *event*.

3. Model Fungsional

   Model fungsional yaitu mengGambarkan transformasi nilai data di dalam sistem *flow diagram* yaitu suatu *graph* dimana *node*-nya menyatakan proses dan *arc*-nya adalah aliran data.

Untuk mempermudah dalam pemodelan abstrak dan pembuatan perancangan program maka digunakan diagram obyek untuk melengkapi notasi grafik dalam pemodelan obyek, kelas dan relasinya dengan yang lain.

Setiap objek mempunyai identitas yang dapat diukur dan memiliki nilai yang bertujuan untuk membedakan entitas antara satu objek dengan objek lain. Pada objek terdapat sifat konkrit yang melekat pada identitas objek tersebut yang berfungssi untuk membedakan setiap objek walaupun nilai atributnya hampir sama atau identik [6].

Objek yang terbagi dalam atribut, operasi, metode, hubungan, dan makna yang sama akan membentuk sebuah kelas yang merupakan wadah bagi objek yang dapat digunakan untuk menciptakan objek, atau dengan kata lain suatu kelas objek mengGambarkan kumpulan dari objek yang mempunyai sifat (atribut), perilaku umum (operasi), relasi umum dengan objek lain dan semantik umum. Fungsi kelas objek adalah mengumpulkan data (atribut) dan perilaku (operasi) yang mempunyai struktur data sama ke dalam satu grup.

2.2 Sistem Transisi

Sistem transisi sering digunakan dalam ilmu komputer sebagai model untuk mengGambarkan perilaku sistem. Sistem transisi pada dasarnya adalah *graph* berarah dimana terdiri dari beberapa *node* yang mewakili *state* dan *edge* dari model transisi yaitu mewakili perubahan *state.* Sebuah *state* menjelaskan beberapa informasi tentang sistem pada saat tertentu dari perilakunya. Misalnya, keadaan lampu lalu lintas menunjukan warna lampu saat ini. Seperti halnya pula pada bagian program komputer yang mengGambarkan seluruh variabel program dengan nilai tertentu dari *program counter* yang mengindikasikan *statement* program mana berikutnya yang akan dieksekusi. Dalam rangkaian perangkat keras *synchronous, state* biasanya mewkili nilai saat ini dari *register* bersamaan dengan dengan nilai masukan berbentuk *bit.* Transisi menentukan bagaimana sistem dapat berkembang dari satu *state* ke *state* lain. Dalam kasus lampu lintas menunjukan peralihan dari satu warna ke warna lainnya. Dalam potongan program komputer biasanya berkaitan dengan pelaksanaan dari *statement* dan melibatkan beberapa variabel dan *counter program.* Kasus pada sirkuit perangkat keras *synchronous*, model transisi terjadi pada perubahan register dan output bit pada satu set masukan [7].

Pada beberapa literatur, sistem transisi menggunakan istilah *action* untuk perubahan status dan *atomic proposition* untuk *states.* Nama dari *action* akan digunakan mengGambarkan mekanisme komunikasi diantara beberapa proses yang terjadi. Dalam hal ini menggunakan huruf alphabet yunani untuk pemberian notasi (seperti α, β, dan seterusnya). *Atomic Proposition* digunakan untuk memformulasikan karakter

temporal. Atomic Proposition secara intuitif mengungkapkan fakta sederhana tentang *state* dari sistem yang ada.

Definisi 2.2.1 [7]

Sebuah sistem transisi TS adalah sebuah tuple (S, Act, →, I, AP, L), dimana :

- *S adalah sebuah himpunan dari state*
- *Act adalah sebuah himpunan dari aksi*
- $\rightarrow \subseteq S \times Act \times S$ *adalah sebuah relasi transisi*
- $I \subseteq S$ *adalah sebuah state awal*
- *AP adalah sebuah atomic proposition, dan*
- $L: S \rightarrow 2^{AP}$ *adalah sebuah fungsi label*

TS dikatakan terbatas jika S, Act dan AP terbatas.

2.3 LTL (Linear Temporal Logic)

Untuk sebuah sistem yang reaktif, kebenaran tidak hanya bergantung pada masukan dan keluaran tetapi juga bergantung pada eksekusi dari sistem. Logika temporal merupakan sebuah formalisme untuk mendeskripsikan sebuah spesifikasi pada sistem dimana dapat mengekspresikan properti dalam berupa notasi matematika. Logika temporal sudah dipelajari pada zaman dahulu pada berbagai bidang seperti filsafat. Aplikasinya adalah seperti pada akhir tahun 1970 Pnueli menggunakan untuk memverifikasi sistem komputer yang cukup rumit [7].

Komponen utama dari formula LTL adalah *atomic proposition* (*state* berlabel $a \in AP$), konektor *Boolean* seperti konjungsi $\wedge$ dan negasi $\neg$, dan dua modal temporal dasar $\circ$ (selanjutnya) dan $\cup$ (sampai). *Atomic propostion* $a \in AP$ adalah kepanjangan untuk label *state a* di dalam sistem transisi. Modal $\circ$ adalah merupakan operator awalan unari dan membutuhkan

statement tunggal LTL sebagai sebuah argument. Formula $\circ\varphi$ menunjukan kondisi saat ini, jika φ menunjukan langkah selanjutnya. Modal $\cup$ merupakan operator biner infiks dan membutuhkan dua formula LTL sebagai argumen. Formula $\varphi_1 \cup \varphi_2$ berlaku untuk kejadian saat ini, jika terdapat kejadian di masa depan dimana φ_2 berlaku dan φ_1 berlaku untuk semua kejadian sampai kejadian di masa depan tersebut [7].

Definisi 2.4.1 [7]

Formula LTL tersebut merupakan sekumpulan dari AP (Atomic Proposition) dibentuk berdasarkan tata bahasa berikut ini :

$$\varphi ::= true|a|\varphi_1 \wedge \varphi_2|\neg\varphi|\circ\varphi|\varphi_1 \cup \varphi_2$$

Dimana $a \in AP$.

Formula LTL digunakan untuk properti berbasis *path*. Hal ini berarti *path* dapat memenuhi formula LTL atau tidak. Untuk lebih jelasnya mengenai bagaimana *path* dapat memenuhi formula LTL tersebut, perhatikan proses berikut ini. Pertama, semantik formula LTL φ didefinisikan sebagai sebagai sebuah bahasa $Words(\varphi)$ yang berisi semua kata tak terbatas mencakup alphabet 2^{AP}yang memenuhi . Artinya, untuk setiap formula LTL sebuah properti LT dikaitkan. Kemudian, semantik diperluas untuk menginterpretasikan *path* dan *state* dalam sebuah sistem transisi.

Definisi 2.4.2 [7]

Jika φ *merupakan formula LTL atas AP, maka properti LT yang diinduksi oleh* φ *adalah sebagai berikut,*

$$Words(\varphi) = \{\sigma \in (2^{AP})^{\omega} \mid \sigma \vDash \varphi\}$$

2.4 PROMELA

PROMELA adalah sebuah bahasa untuk membangun model verifikasi yang merepresentasikan suatu abstrak dari

sebuah sistem, yang hanya berisi aspek – aspek yang relevan dengan sifat seseorang ketika ingin memverifikasi [8].

Sebuah program PROMELA terdiri dari proses, saluran pesan (*messages channel),* dan variabel. Proses didefinisikan secara global, sementara saluran pesan dan variabel dapat didefinisikan secara global maupun lokal dalam sebuah proses. Proses yang digunakan untuk menentukan sistem *behavior system, channel* dan variabel global yang digunakan untuk mendefinisikan lingkungan dimana proses berjalan [1].

PROMELA adalah bahasa yang dirancang untuk mengGambarkan sistem yang terdiri dari konkuren *asynchronous* dalam proses komunikasi [1].

Hal tersebut terdiri dari satu set pemrosesan berhingga, saluran global dan lokal, dan variabel global dan lokal. Proses komunikasi melalui pesan melalui saluran. Komunikasi *asynchronous* menggunakan saluran sebagai buffer yang berhingga, dan komunikasi *synchronous* menggunakan saluran dengan ukuran nol. Saluran dan variabel global menentukan lingkungan dimana proses tersebut berjalan, sementara saluran dan variabel lokal untuk menentukan *internal local state* dari pemrosesan.

PROMELA adalah bahasa non – deterministik yang meminjam beberapa konsep dan elemen sintaks dari *Dijkstra's guarded command language*. Proses PROMELA didefinisikan sebagai barisan kalimat yang mungkin berlabel yang didahului oleh bagian deklaratif. Kalimat dasar dalam PROMELA adalah sesuatu yang menghasilkan efek yang pasti dalam *model state,* dengan kata lain instruksi untuk mengirim dan menerima pesan ke (dari) saluran dan ekspresi *Boolean*, BExp mencakup tes lebih variabel dan isinya [3].

2.5 SPIN (*Simple Promela Interpreter*)

SPIN adalah sistem verifikasi generik yang mendukung desain dan verifikasi sistem proses *asynchronous*. Proses *asynchronous* adalah kendali yang akan kembali kepada pengguna tanpa menunggu proses masukan dan keluaran selesai sehingga tidak terjadi *wait loop* atau waktu tunggu. Model verifikasi SPIN terfokus pada pembuktian kebenaran dari proses interaksi dan mencoba dari abstraksi sistem sedemikian mungkin dari barisan komputasinya [9].

Secara *default,* diberikan formula LTL, SPIN melakukan penerjemahan ke dalam automata yang merupakan perilaku yang tidak diinginkan atau diklaim tidak mungkin. Kemudian, verifikasi terdiri dari eksplorasi lengkap dari ruang *state* untuk eksekusi yang memenuhi automata tersebut. Jika eksekusi tersebut ada, maka *tools report* itu sebagai *counter example* untuk properti. Jika model ini dieksplorasi dan *counter example* tidak ditemukan, maka model memenuhi properti LTL sebagai properti universal. Skema verifikasi yang sama dapat digunakan untuk memeriksa apakah formula tidak dapat dipenuhi oleh *path* apapun (sanggahan dari properti eksistensial). Kedua, cara menggunakan LTL disajikan dalam *interface* yang *user friendly* disebut dengan JSpin [9].

Meskipun ada banyak contoh nyata dimana verifikasi bisa dilakukan dengan verifikasi lengkap standar, SPIN juga menerapkan teknik optimasi untuk menangani sistem yang kompleks. Pengurangan rangka parsial menggantikan beberapa barisan [8].

Untuk memeriksa kebenaran sistem dengan logika properti sistem yang ditentukan dalam *linear temporal logic,* SPIN pertama mengubah formula menjadi tes automata yang bekerja

seperti pengamat atau monitor dalam sebuah sistem. Sambil membangun sistem eksekusi, monitor melakukan konsultasi untuk setiap langkahnya untuk melihat apakah terjadi pelanggaran. Jika pelanggaran terdeteksi, SPIN menampilkan barisan yang *interleaving* tepat dari *state* awal ke *state* yang dimana terjadi pelanggaran. Ini berfungsi sebagai contoh kontra untuk kebenaran yang mengklaim dan memfasilitasi diagnosis deteksi pelanggaran [9].

BAB III

METODOLOGI PENELITIAN

Dalam bab ini, diuraikan langkah – langkah dalam penelitian tugas akhir ini yaitu studi literatur, mendefinisikan model bisnis sistem ATM dalam bentuk diagram transisi, mendefinisikan model diagram transisi dalam PROMELA, mendefinisikan property LTL, verifikasi dan penarikan kesimpulan beserta pembukuan tugas akhir.

1. Studi Literatur.

Pada tahap ini telah dilakukan pengkajian mengenai pemodelan berorientasi obyek, sistem transisi, model EFSM *(Extended Finite State Machine)*, LTL (*Linear Temporal Logic),* PROMELA dan SPIN (*Simple Promela Interpreter*). Pengkajian sistem transisi dan model EFSM bersumber pada jurnal yang berjudul *A case Study of Model Checking Retail Banking* dan *A Tool for Abstraction in Model Checking*. Sedangkan, untuk mengkaji mengenai LTL, PROMELA dan SPIN, penulis menggunakan beberapa jurnal dan buku, diantaranya adalah *Principles of Model Checking, SPIN Model Checking and Software* dan *Object Oriented Modeling and Design*.

Selain melakukan pengkajian pada beberapa referensi, penulis juga melakukan analisa untuk mendeskripsikan model bisnis sistem ATM, dan dalam hal ini penulis melakukan analisa menggunakan bantuan model bisnis yang ada pada buku *Object Oriented Modeling and Design*.

Dalam buku tersebut, dijabarkan mengenai model bisnis sistem ATM serta diagram interaksi. Dalam hal ini, penulis berniat untuk melakukan verifikasi terhadap model yang ada.

2. Mendefinisikan Model Bisnis Sistem ATM dalam Bentuk Sistem Transisi.

Pada tahap ini, dilakukan proses analisa terhadap model bisnis sistem ATM. Model sistem yang diberikan adalah dalam bentuk model berorientasi obyek, dimana penulis mengambil model sistem tersebut pada buku *Object Oriented Modeling and Design*. Analisa dilakukan untuk melakuakn transformasi sistem dari model dalam diagram alir tersebut ke dalam sistem transisi.

Transformasi dari model diagram alir ke sistem transisi dibutuhkan beberapa pengetahuan mengenai model berorientasi obyek. Karena dalam model tersebut mengandung beberapa informasi yang dapat dibaca jika penulis mampu memahami model tersebut.

Analisa terhadap proses bisnis dilakukan dalam tahap ini, sehingga *output* dari tahap ini adalah pendefinisian terhadap model yang kemudian diubah menjadi sistem transisi, dimana nantinya akan dilakukan pendefinisian menjadi bahasa PROMELA.

Tujuan diubahnya menjadi sistem transisi adalah untuk mempermudah menganalisa serta pendefinisian terhadap bahasa PROMELA dimana nantinya akan dilakukan proses verifikasi. Oleh sebab itu, tahap ini

merupakan tahap yang cukup penting dan sangat krusial.

3. Mendefinisikan Model Sistem Transisi dalam PROMELA.

Setelah mendapatkan bentuk sistem transisi dari model bisnis ATM tersebut. Langkah selanjutnya, adalah menfenisikan model diagram transis tersebut dalam bahasa PROMELA.

Untuk mendefiniskan diagram transisi dalam bahasa PROMELA, perlu dilakukan beberapa analisa yaitu, penentuan *proctype, mtype*, *channel*, *Boolean*, *class* atau entitas dan *atomic process*. Selain itu, tidak kalah penting adalah penentuan *state* dimana nantinya *state* tersebut akan dimasukan ke dalam beberapa *mtype*.

Langkah pertama yang harus dilakukan adalah melakukan klasifikasi proses dimana hal tersebut nantinya akan membentuk *proctype*. Setelah itu, adalah melakukan generalisasi terhadap masing – masing proses untuk diketahui aksi apa saja yang ada pada setiap proses, hal ini nantinya akan membentuk *mtype*. Kemudian, setelah terbentuk beberapa *mtype,* aksi – aski pada masing – masing *mtype* harus diperjelas dan dibentuk menjadi sebuah kata khusus tanpa spasi, dimana hal itu diperoleh dari *state* yang ada dalam diagram sistem transisi. Terakhir, dibentuk beberapa *channel*, dimana hal ini didasarkan pada banyak *proctype* dan *mtype*, serta relasi diantara masing – masing proses pada *proctype*.

4. **Mendefinisikan Properti LTL.**

Dalam tahap ini, didefinisikan properti LTL (*Linear Time Logic)* untuk proses verifikasi. Properti yang akan dibangun meliputi *liveness* dan *safety.*

Sebelum mendefinsikan property tersebut,perlu diuraikan detil dari permasalahan yang akan dibahas. Kemudian properti tersebut akan diubah menjadi LTL agar dapat dilakukan proses verifikasi dengan SPIN.

5. **Verifikasi dengan SPIN.**

Setelah PROMELA dan property selesai dikerjakan, maka langkah selanjutnya adalah melakukan proses verifikasi. Dimana dalam proses ini, akan dilakukan analisa terkait apakah spesifikasi yang diberikan dapat memenuhi sistem yang ada. Jika ada spesifikasi yang tidak terpenuhi, maka sistem perlu diperbaiki.

6. **Penarikan Kesimpulan dan Pembukuan Tugas Akhir.**

Setelah melakukan penelitian, dibuatlah kesimpulan – kesimpulan berdasarkan penelitian yang telah dilakukan. Kemudian, disusun buku tugas akhir yang menjelaskan mengenai penelitian yang telah dilakukan berdasarkan sistematika penulisan buku tugas akhir yang ada.

BAB IV

ANALISIS DAN PEMBAHASAN

4.1 Mode Simulasi dalam SPIN

SPIN mendukung tiga mode simulasi, diantaranya :

1. Simulasi acak
2. Simulasi terarah
3. Simulasi lengkap dengan *Depth First Search*

Untuk menentukan model simulasi untuk digunakan dalam tugas akhir ini, penulis menulis sebuah program uji seperti pada Gambar 4.1 dan menjalankannya melalui SPIN.

```
chan reader = [1] of {int};

proctype setReader(){
    reader!1;
}
proctype unsetReader(){
    reader!0;
}
init {
    run setReader();
    run unsetReader();
    int x;
    reader?x;
    printf("%d\n", x);
    assert(x==1);
}
```

Gambar 4. 1 Mode Simulasi SPIN

4.1.1 Simulasi Acak

Dalam simulasi acak, SPIN secara acak memilih cabang pada titik pilihan pada pohon perhitungan, seperti yang diGambarkan di bawah ini. Ketika menjalankan SPIN pada program uji tersebut, ditemukan bahwa terdapat dua pernyataan yang dieksekusi secara acak. Terdapat pelanggaran yang terjadi dalam simulasi tersebut dapat dilihat pada Gambar 4.2 dengan memperhatikan kalimat yang telah disorot warna kuning. Karena perilaku ini, simulasi acak tidak cocok untuk proses verifikasi dalam peneilitian ini, karena mengeksplorasi hanya satu jejak eksekusi.

```
  0:    proc  - (:root:) creates proc  0 (:init:)
Starting setReader with pid 1
  1:    proc  0 (:init:) creates proc  1 (setReader)
0 :init ini run setReader(
Starting unsetReader with pid 2
  2:    proc  0 (:init:) creates proc  2 (unsetReader)
0 :init ini run unsetReade
2 unset 7   values: 1!0
2 unset 7   reader!0
Process Statement          reader
0 :init ini values: 1?0    [0]
0 :init ini reader?x       [0]
  4:    proc  2 (unsetReader) terminates
Process Statement          :init:(0): reader
1 setRe 4   values: 1!1    0
1 setRe 4   reader!1       0
  5:    proc  1 (setReader) terminates
0
0 :init ini printf('%d\\n' 0          [1]
spin: test simulation:15, Error: assertion violated
spin: text of failed assertion: assert((x==1))
#processes: 1
  7:    proc  0 (:init:) test_simulation:15 (state 5)
3 processes created
```

Gambar 4. 2 Hasil Mode Simulasi Acak dalam SPIN

4.1.2 Simulasi Terarah

Dalam simulasi terarah, SPIN berjalan dalam mode interaktif. Pada setiap titik pilihan, SPIN meminta pengguna untuk memilih cabang untuk transit. Dalam hal ini, SPIN berjalan dengan mudah pada mode ini pada program uji tersebut karena poin pilihan terbatas. Namun, dapat dibayangkan bahwa jumlah poin pilihan untuk yang akan dilintasi pada sistem ATM akan sangat kompleks sehingga dalam praktiknya simulasi terarah tidak dapat digunakan. Dan untuk mengetahui mana cabang yang dipilih mengharuskan pengguna sudah tahu mengenai kesalahan yang dapat terjadi, dan ini tidak cocok untuk sistem ATM. Simulasi terarah hanya berguna untuk menjelajahi beberapa jejak eksekusi yang diketahui.

4.1.3 Simulasi *Exhaustive Depth First Search*

Dalam simulasi *Exhaustive Depth First Search*, SPIN memilih transisi yang belum dijelajahi dan akan kembali untuk menjelajahi yang lain. Ketika SPIN telah selesai dengan satu subtree dari *computational tree*, maka SPIN terus berlanjut pada *sibling*, sampai seluruh *computational tree* dijelajahi. Keuntungan dari mode ini adalah bahwa hal ini otomatis dan mencakup semua kemungkinan jejak eksekusi untuk memvalidasi properti. Pada simulasi tersebut tidak ditemukan pelanggaran ataupun eror, ditunjukan pada Gambar 4.3.

```
 0:    proc  - (:root:) creates proc  0 (:init:)
Starting setReader with pid 1
  1:   proc  0 (:init:) creates proc  1
(setReader)
0 :init ini run setReader(
1 setRe 4   values: 1!1
1 setRe 4   reader!1
Starting unsetReader with pid 2
  3:   proc  0 (:init:) creates proc  2
(unsetReader)
Process Statement          reader
0 :init ini run unsetReade [1]
0 :init ini values: 1?1    [1]
0 :init ini reader?x       [1]
Process Statement          :init:(0): reader
2 unset 7   values: 1!0    1
2 unset 7   reader!0       1
  5:   proc  2 (unsetReader) terminates
  5:   proc  1 (setReader) terminates
1
0 :init ini printf('%d\\n' 1          [0]
0 :init ini assert((x==1)) 1          [0]
  7:   proc  0 (:init:) terminates
3 processes created
```

Gambar 4. 3 Hasil Mode Simulasi *Exhaustive Depth First Search* dalam SPIN

4.1.4 Mode Simulasi yang digunakan

Dalam tugas akhir ini, mode simulasi harus menjelajahi semua jejak eksekusi yang mungkin untuk memvalidasi properti yang penting. Dalam hal ini tidak mungkin menjelajahi beberapa jejak saja untuk menyelediki semua properti atau mengetahui jejak mana saja yang harus ditelusuri, karena akan dipilih secara acak. Oleh karena itu, penulis telah memutuskan untuk menggunakan *Exhaustive Depth First Search* terhadap kode Promela yang telah ditulis pada Tugas Akhir ini.

4.2 Asumsi Proses Bisnis Sistem ATM

Mengingat kendala waktu dan kompleksitas mesin ATM, penulis telah membuat asumsi berikut untuk menyederhanakan model ATM kami.

1. Apabila pengguna memasukkan pin yang salah, ATM akan meminta Pengguna masukkan kembali pin. Berdasarkan diagram *state*, hal ini akan menyebabkan perulangan tak terbatas antara pengguna dan ATM jika pin salah dimasukkan berulang kali. Pada kenyataannya, sebagian besar ATM akan menelan kartu setelah 3 kali gagal. Untuk menyederhanakan program tersebut, setelah pin yang salah dimasukkan, ATM akan menampilkan pesan sandi yang salah dan mengeluarkan kartu.
2. Apabila terjadi gagal transaksi, model dari *Object-Oriented Modeling and Design* [6] akan menunggu 5 detik, kemudian menampilkan pesan sukses dan coba lagi dengan meminta pengguna untuk menentukan transaksi yang akan dilakukan. Untuk menyederhanakan program tersebut, model ini akan melewatkan menunggu 5 detik tapi akan menampilkan pesan sukses dan meminta pengguna untuk menentukan transaksi yang akan dilakukan.
3. Model dari *Object-Oriented Modeling and Design* [6] akan memverifikasi akun setelah pengguna memasukkan pin yang benar. Jika akun tersebut adalah akun yang rusak (misalkan akun yang sudah kadaluwarsa), ATM akan menampilkan pesan akun rusak, mencetak tanda terima dan mengeluarkan kartu. Untuk menyederhanakan program ini, model kami akan melewati verifikasi akun setelah pengguna memasukkan pin yang benar.

4. Diagram interaksi dari *Object-Oriented Modeling and Design* [6] hanya mengGambarkan skenario transaksi tertentu dari ATM yaitu, penarikan uang tunai berhasil. Padahal ada banyak kemungkinan transaksi dari sistem ATM yang dapat dieksekusi, misalnya gagal penarikan uang, gagal atau sukses Pemeriksaan saldo akun atau pengguna membatalkan transaksi pada titik-titik yang berbeda dari transaksi, seterusnya dan sebagainya. Untuk tujuan studi kasus dalam hal ini akan dibatasi pemodelan kita tentang sistem ini hanya untuk skenario transaksi berikut:
   - Penarikan uang berhasil
   - Pemeriksaan saldo berhasil
   - Pengguna memasukkan kartu tak terbaca
   - Pengguna memasukkan pin yang salah
   - Pengguna membatalkan transaksi setelah diminta untuk memasukan pin
   - Pengguna membatalkan transaksi setelah diminta untuk menentukan transaksi yang akan dilakukan
   - Pengguna membatalkan transaksi setelah diminta untuk menentukan jumlah yang akan ditarik
   - Transaksi gagal dari konsorsium
   - Transaksi gagal dari bank

4.3 Pemodelan Sistem ATM dalam Diagram Transisi

Dalam proses bisnis ATM, terdapat beberapa aktivitas diantaranya adalah menyisipkan kartu, memasukan pin / password, memasukan jumlah uang, mengambil uang, mengambil kartu, membatalkan transaksi, melanjutkan transaksi, memproses transaksi, menampilkan tampilan utama, menamilkan pesan kartu yang tak terbaca, meminta password, menampilkan pesan pembatalan, mengeluarkan kartu, mencetak tanda bukti, verifikasi password oleh Bank, verifikasi transaksi

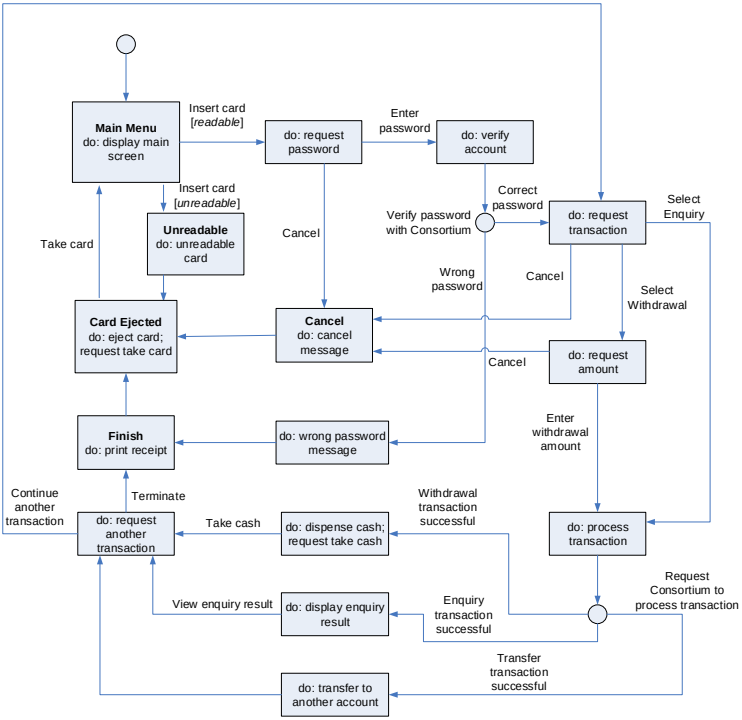
proses oleh Bank dan Konsorsium, melakukan pemeriksaan saldo dan pengambilan uang. Semua aktivitas tersebut akan disusun menjadi sebuah skenario, dimana terdapat dua skenario yaitu skenario normal pada ATM dan skenario ATM dengan pengeceualian.

Skenario normal merupakan simulasi aktivitas pada sistem ATM secara normal dengan mengakses keseluruhan *state* tanpa adanya pembatalan ataupun validasi yang gagal. ATM mula – mula meminta pengguna memasukan kartu, pengguna memasukan kartu. ATM menerima kartu dengan membaca nomor serial pada kartu. Kemudian, pengguna diminta untuk memasukan password, pengguna memasukan password, sebagai contoh adalah "123456". ATM memeriksa password dan juga nomor serial pada kartu, dilakukan oleh konsorsium, Konsorsium melakukan pemeriksaan ini bersama dengan Bank dan memberikan notifikasi kepada ATM bahwa kartu diterima. Setelah kartu diterima, pengguna diminta untuk memasukan jenis transaksi (pengambilan uang tunai atau cek saldo), dalam kasus ini diasumsikan pengguna melakukan pengambilan uang tunai. ATM meminta pengguna memasukan jumlah uang yang akan diambil. ATM akan melakukan verifikasi mengenai batas uang yang dimiliki oleh pengguna melalui konsorsium, kemudian konsorsium bersama dengan Bank memberika notifikasi kepada ATM mengenai informasi saldo pengguna dan tindakan selanjutnya apakah pengguna dapat melakukan penarikan uang (saldo cukup) ataukah penolakan transaksi (saldo tidak cukup). Dalam kasus ini, diasumsikan bahwa saldo cukup, maka uang akan dikeluarkan oleh ATM. Kemudian, ATM akan bertanya kepada pengguna apakah ingin melanjutkan transaksi atau selesai. Jika pengguna melanjutkan transaksi maka pengguna akan kembali pada permintaan transaksi yang dilakukan oleh konsorsium dan Bank. Jika pengguna tidak melanjutkan transaksi maka, ATM mengeluarkan tanda bukti. Setelah itu, ATM akan mengeluarkan kartu pengguna. Dan terakhir, pengguna diminta

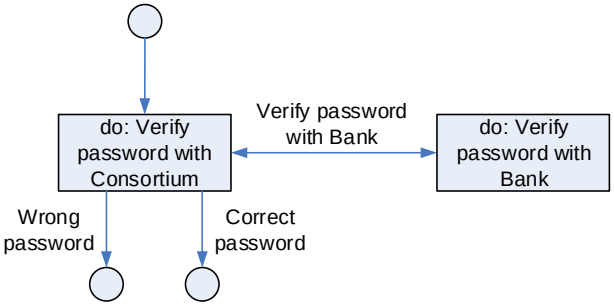
kembali untuk memasukan kartu jika ingin memulai proses dari awal.

Skenario dengan pengecualian merupakan simulasi aktivitas pada sistem ATM dengan adanya beberapa pengecualian yang terjadi karena pengguna memilih untuk tidak melakukan transaksi secara normal. ATM meminta pengguna memasukan kartu dan pengguna memasukan kartu. ATM menelan kartu dan membaca nomor serial yang ada pada kartu. ATM meminta pengguna memasukan password, dan pengguna memasukan password, sebagai contoh pengguna memasukan "123456". Kemudian, ATM melakukan verifikasi terhadap password dan nomor serial kartu bersama dengan konsorsium dan Bank. Kemudian, ATM mengindikasikan adanya password yang tidak benar karena notifikasi yang dikirimkan oleh Bank dan Konsorsium. Maka, pengguna diminta memasukan kembali passwordnya. Jika sudah benar, maka pengguna akan masuk ke tampilan utama ATM. Pengguna memilih menu pengambilan uang tunai, tetapi pengguna berubah pikiran dan menekan tombol "batal" untuk membatalkan transaksi. Sehingga transaksi dibatalkan dan kartu dikeluarkan oleh ATM.

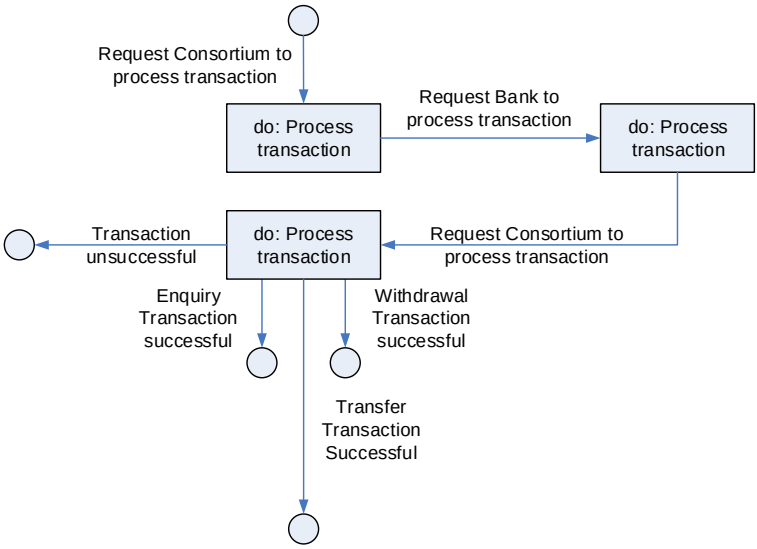
Kedua skenario tersebut, diasumsikan untuk mengetahui proses bisnis ATM yang akan dibangun pada diagram transisi. Selain itu, dapat juga kita mengetahui mengenai komunikasi antara pengguna, ATM, konsorsium dan Bank. Berdasarkan skenario diatas telah disusun diagram transisi pada Gambar 4.4, 4.5 dan 4.6.



Gambar 4. 4 Diagram Transisi Sistem ATM



Gambar 4. 5 Diagram Transisi Sistem ATM Proses Verifikasi *Password*



Gambar 4.6 Diagram Transisi Sistem ATM Proses Transaksi

Berdasarkan proses bisnis pada Gambar 4.4 – 4.6, dibuat empat proses untuk model transaksi, empat proses akan mewakili empat entitas aktif yang ada di dalam sistem ATM, yaitu :

- Pengguna
- ATM
- Konsorsium
- Bank

Setiap proses ditulis dengan *if statement* dengan kurung kurawal. Sebuah potongan proses ATM ditunjukan pada Gambar 4.7.

```
proctype ATM(){
       printf("initiating ATM Process...\n");
       DisplayMenuState:
       atm2user!DISPLAY_MAIN_MENU ->
              printf("ATM       Process:       Main
Menu...\n");
       user2atm?INSERT_CARD ->
              goto ReadCardState;

       ReadCardState:
       if
       ::     printf("ATM    Process:    Card    is
valid...\n");
              atm2user!VALID_CARD;
              printf("ATM Process: Prompt user for
password...\n");
              atm2user!REQUEST_PASSWORD;
              goto ReceiveAccState;
       ::     printf("ATM    Process:    Card    is
unreadable...\n");
              atm2user!INVALID_CARD;
              atm2user!REQ_REMOVE_CARD;
              goto EjectCardState;
       fi;

       ...

}
```

Gambar 4. 7 Proses ATM dalam PROMELA

Aksi yang berbeda menyatakan bahwa entitas yang berbeda yang mampu diambil dapat diwakili oleh 4 jenis dalam model Promela berikut ini :

Aksi pada ATM:

- DISPLAY_MAIN_MENU
- VALID_CARD
- INVALID_CARD
- REQUEST_PASSWORD
- VERIFY_PW_CON

- REQ_TRANSACTION
- REQ_AMT
- REQ_AMT_TRF
- ATM_PROCESS_TRANS
- REQ_CONTINUE
- REQ_REMOVE_CARD

Aksi untuk Pengguna :

- INSERT_CARD
- ENTERED_PASSWORD
- CANCEL
- WITHDRAW
- ENQUIRY
- TRANSFER
- ENTERED_AMT
- TOOK_CASH
- TERMINATE
- CONTINUE
- REMOVED_CARD

Aksi untuk Konsorsium:

- VERIFY_PW_BANK
- CORRECT_PW
- INCORRECT_PW
- CONSORTIUM_PROCESS_TRANS
- TRANSACTION_SUCCESS
- TRANSACTION_UNSUCCESS
- ANOTHER_TRANSACTION
- END_TRANSACTION

Aksi untuk Bank:

- VERIFIED_PW
- WRONG_PW
- BANK_TRANS_SUCCESS
- BANK_TRANS_FAIL
- ANOTHER_TRANSACTION
- END_TRANSACTION

Berdasarkan aksi – aksi yang telah dibuat maka akan dilakukan proses peng-kode-an dalam PROMELA. Kode promela untuk 4 jenis mtypes untuk mewakili aksi yang ditunjukkan pada Gambar 4.8.

```
mtype = {DISPLAY_MAIN_MENU, VALID_CARD,
INVALID_CARD, REQUEST_PASSWORD, VERIFY_PW_CON,
REQ_TRANSACTION, REQ_AMT, ATM_PROCESS_TRANS,
REQ_CONTINUE, REQ_REMOVE_CARD}; /* ATM action */

mtype = {INSERT_CARD, ENTERED_PASSWORD, CANCEL,
WITHDRAW, ENQUIRY, TRANSFER, ENTERED_AMT,
ENTERED_AMT_TRF TOOK_CASH, TERMINATE, CONTINUE,
REMOVED_CARD};

mtype = {VERIFY_PW_BANK, CORRECT_PW, INCORRECT_PW,
CONSORTIUM_PROCESS_TRANS, TRANSACTION_SUCCESS,
TRANSACTION_UNSUCCESS};

mtype = {VERIFIED_PW, WRONG_PW,
BANK_TRANS_SUCCESS, BANK_TRANS_FAIL,
ANOTHER_TRANSACTION, END_TRANSACTION};
```

Gambar 4.8 *mtype* Sistem ATM dalam PROMELA

Dari proses bisnis tersebut, dapat diperoleh informasi mengenai saluran komunikasi yang terjadi pada sistem ATM. Enam saluran

akan dibuat untuk mengelola komunikasi antara entitas yang berbeda yaitu sebagai berikut :

- **user2atm**: saluran komunikasi untuk pengguna ke ATM
- **atm2user**: saluran komunikasi dari ATM ke pengguna
- **atm2consortium**: saluran komunikasi dari ATM ke Konsorsium
- **consortium2atm**: saluran komunikasi dari Konsorsium to ATM
- **consortium2bank**: saluran komunikasi dari Konsorsium ke Bank
- **bank2consortium**: saluran komunikasi dari Bank ke Konsorsium

6 saluran akan didefinisikan sebagai elemen *mtype* dengan panjang 1 (seperti yang ditunjukkan di bawah). elemen *mtype* yang ditemukan pada saluran pada contoh spesifik yang akan diwakili oleh *state* bahwa 4 entitas yang berbeda ada di dalam. Proses pengkodean dalam PROMELA ditunjukan pada Gambar 4.9.

```
chan user2atm = [1] of {mtype};
chan atm2user = [1] of {mtype};
chan atm2consortium = [1] of {mtype};
chan consortium2atm = [1] of {mtype};
chan consortium2bank = [1] of {mtype};
chan bank2consortium = [1] of {mtype};
```

Gambar 4. 9 *channel* Sistem ATM dalam PROMELA

Program Promela diinisialisasi dengan menjalankan semua empat proses utama seperti pada Gambar 4.10.

```
init {
       atomic {
       run ATM();
       run User();
       run Consortium();
       run Bank();
       }
}
```

Gambar 4. 10 Inisialisasi Program PROMELA

4.4 Pemodelan Sistem ATM dalam PROMELA

4.4.1 Definisi *mtype, channel* dan *boolean*

Pada subbab ini dibahas mengenai pendefinisian tiga unsur utama dalam PROMELA yaitu, *mtype, channel* dan b*oolean.* Dimana pada subbab sebelumnya telah disinggung mengenai *mtype* dan *channel*.

Dalam tugas akhir ini, *mtype* dibagi menjadi 4 bagian yaitu, pengguna, ATM, konsorsium dan Bank. Isi dari *mtype* adalah aksi – aksi yang akan dilakukan oleh masing – masing *role*. Dari empat *role* yang ada pastinya minimal akan terdapat 3 pasang saluran pesan atau *message channel*.

Message channel tersebut merupakan perwakilan dari sekumpulan *mtype* yang saling berinteraksi. Terdapat 6 buah *message channel* dimana telah dijelaskan pada subbab sebelumnya.

Dalam proses pemeriksaan PROMELA hanya menggunakan dua unsur yaitu *mtype* dan *message channel*. Sehingga, terlihat *Boolean* tidak memiliki peran. Tetapi setelah dilakukan *import* untuk spesifikasi / properti LTL maka *Boolean* memiliki peran, karena dalam LTL hanya *Boolean* yang digunakan. Dan *Boolean* hanya memiliki nilai *true* atau *false.* Dimana dalam tugas akhir ini diberikan inisialisasi awal untuk semua *boolean* adalah *false.* Masing – masing *boolean* telah diuraikan pada Tabel 4.1. Urutan penulisan yang benar adalah

mtype, kemudian diikuti oleh *message channel* dan terakhir adalah *boolean,* seperti pada Gambar 4.11

Tabel 4.1 Boolean dalam Kode PROMELA ATM

Boolean	**Makna**
user_cancel	Apakah pengguna menekan tombol batal ?
password_correct	Apakah pin benar ?
withdraw_selected	Apakah penarikkan dipilih ketika pengguna diminta untuk memilih jenis transaksi?
enquiry_selected	Apakah informasi saldo yang dipilih ketika pengguna diminta untuk memilih jenis transaksi?
transfer_selected	Apakah transfer yang dipilih ketika pengguna diminta untuk memilih jenis transaksi?
receipt_printed	Apakah tanda terima yang dicetak?
card_ejected	Apakah kartu dikeluarkan?
cash_dispensed	Apakah uang tunai tersedia ?
transaction_success	Apakah transaksi berhasil sebagai feedback dari konsorsium?
amount_entered	Berapakah jumlah penarikan yang dimasukkan oleh pengguna?

Continue_transaction	Apakah akan melanjutkan ke transaksi lain ?

```
mtype     =     {DISPLAY_MAIN_MENU,     VALID_CARD,
INVALID_CARD,   REQUEST_PASSWORD,   VERIFY_PW_CON,
REQ_TRANSACTION,    REQ_AMT,    ATM_PROCESS_TRANS,
REQ_CONTINUE, REQ_REMOVE_CARD};  /* ATM action */
mtype  =  {INSERT_CARD,  ENTERED_PASSWORD,  CANCEL,
WITHDRAW,    ENQUIRY,    TRANSFER,    ENTERED_AMT,
ENTERED_AMT_TRF  TOOK_CASH,  TERMINATE,  CONTINUE,
REMOVED_CARD};
mtype = {VERIFY_PW_BANK, CORRECT_PW, INCORRECT_PW,
CONSORTIUM_PROCESS_TRANS,     TRANSACTION_SUCCESS,
TRANSACTION_UNSUCCESS};
mtype       =       {VERIFIED_PW,       WRONG_PW,
BANK_TRANS_SUCCESS,              BANK_TRANS_FAIL,
ANOTHER_TRANSACTION, END_TRANSACTION};

chan user2atm = [1] of {mtype};
chan atm2user = [1] of {mtype};
chan atm2consortium = [1] of {mtype};
chan consortium2atm = [1] of {mtype};
chan consortium2bank = [1] of {mtype};
chan bank2consortium = [1] of {mtype};

bool user_cancel = false;
bool password_correct = false;
bool withdraw_selected = false;
bool enquiry_selected = false;
bool transfer_selected = false;
bool receipt_printed = false;
bool card_ejected = false;
bool cash_dispensed = false;
bool transaction_success = false;
bool amount_entered = false;
bool continue_transaction = false;
```

Gambar 4.11 Definisi *mtype, channel* dan *boolean*

4.4.2 Definisi *proctype Consortium*

Konsorsium merupakan sebuah entitas yang memiliki interaksi dengan ATM dan Bank. Konsorsium memiliki peran untuk melakukan verifikasi password dan pengaturan dalam proses transaksi.

Pada proses verifikasi password, konsorsium mempunyai 4 *state*, yaitu ATM melakukan permohonan agar konsorsium melakukan verifikasi password, Konsorsium melakukan verifikasi password dengan memberikan nilai benar yang dikomunikasikan dengan Bank, Konsorsium melakukan verifikasi password dengan memberikan nilai salah yang dikomunikasikan dengan Bank, dan ATM selesai melakukan komunikasi dengan konsorsium setelah menerima nilai yang diberikan oleh Bank melalui Konsorsium.

Pada pengaturan proses transaksi, konsorsium memiliki 6 *sate*, yaitu ATM melakukan permohonan kepada konsorsium untuk melakukan transaksi, Bank melakukan persetujuan melalui konsorsium kepada ATM bahwa permohonan yang dilakukan oleh ATM sebelumnya disetujui, Bank melakukan persetujuan melalui konsorsium kepada ATM bahwa permohonan yang dilakukan oleh ATM sebelumnya tidak disetujui atau gagal, ATM melakukan permohonan untuk menyelesaikan transaksi kepada konsorsium dan ATM melakukan permohonan untuk melakukan transaksi lainnya kepada konsorsium.

Dua proses tersebut didefinisikan ke dalam bahasa PROMELA dalam sebuah rangkaian *if statement* seperti pada Gambar 4.12.

```
proctype Consortium(){
printf("initiating Consortium Process...\n");
VerifyPassword:
       if
:: atm2consortium?VERIFY_PW_CON ->
       printf("Consortium    Process:    consortium
       verifying acc with bank...\n");
       consortium2bank!VERIFY_PW_BANK;
       if
:: bank2consortium?VERIFIED_PW ->
       consortium2atm!CORRECT_PW;
       printf("Consortium    Process:    consortium
       verified acc: correct password...\n");
       goto ProcessTransaction;
:: bank2consortium?WRONG_PW ->
       consortium2atm!INCORRECT_PW;
       printf("Consortium    Process:    consortium
       verified acc: incorrect password...\n");
       goto VerifyPassword
       fi;
:: atm2consortium?END_TRANSACTION ->
       consortium2bank!END_TRANSACTION;
       goto end;
       fi;
ProcessTransaction:
       if
:: atm2consortium?ATM_PROCESS_TRANS ->
       printf("Consortium    Process:    consortium
       process transaction...\n");
       consortium2bank!CONSORTIUM_PROCESS_TRANS;
       if
:: bank2consortium?BANK_TRANS_SUCCESS ->
       consortium2atm!TRANSACTION_SUCCESS;
       printf("Consortium    Process:    consortium
       processed transaction successfully...\n");
:: bank2consortium?BANK_TRANS_FAIL ->
       consortium2atm!TRANSACTION_UNSUCCESS;
       printf("Consortium   Process:   transaction
       failed...\n");
       fi;
```

```
:: atm2consortium?END_TRANSACTION ->
       consortium2bank!END_TRANSACTION;
       goto end;
       fi;
       if
:: atm2consortium?ANOTHER_TRANSACTION ->
       consortium2bank!ANOTHER_TRANSACTION;
       goto ProcessTransaction;
:: atm2consortium?END_TRANSACTION ->
       consortium2bank!END_TRANSACTION;
       goto end;
       fi;
end:
       printf("Consortium Process: ending ...\n");
}
```

Gambar 4.12 Definisi *proctype* Consortium

4.4.3 Definisi *proctype Bank*

Bank merupakan sebuah entitas yang memiliki interaksi dengan konsorsium. Bank memiliki peran untuk melakukan verifikasi password dan pengaturan dalam proses transaksi.

Pada proses verifikasi password, konsorsium mempunyai 4 *state*, yaitu Konsorsium memberikan informasi kepada Bank mengenai akun pengguna yang dikirimkan melalui ATM sehingga Bank dapat melakukan verifikasi terhadap akun tersebut. Kedua, Jika akun berhasil terverifikasi oleh Bank maka Konsorsium akan mengrimkan sebuah pesan kepada ATM bahwa ATM dapat melanjutkan transaksi berikutnya. Ketiga, jika akun tidak berhasil diverifikasi, maka konsorsium akan memberikan informasi kepada ATM untuk melakukan verifikasi password kembali. Terakhir, Bank akan memberikan informasi kepada konsorsium untuk menyelesaikan proses transaksi.

Pada pengaturan proses transaksi, konsorsium memiliki 6 *state*, yaitu konsorsium memberikan informasi dari ATM untuk meminta izin bahwa ATM akan melakukan transaksi. Kedua, Bank memberikan informasi kepada konsorsium bahwa transaksi

berhasil dilakukan, sehingga transaksi akan diselesaikan. Ketiga, apabila transaksi gagal dilakukan, maka transaksi juga harus diselesaikan. Keempat, ketika konsorsium meminta informasi kepada Bank bahwa akan melanjutkan ke transaksi lainnya, maka akan dilanjutkan ke state ProcessTransaction. Dan terakhir, apabila konsorsium meminta Bank untuk mengakhiri transaksi maka Bank akan mengirimkan informasi bahwa proses berakhir.

Dua proses tersebut didefinisikan ke dalam bahasa PROMELA dalam sebuah rangkaian *if statement* seperti pada Gambar 4.13

```
proctype Bank(){
        printf("initiating Bank Process...\n");
        VerifyPassword:
        if
::  consortium2bank?VERIFY_PW_BANK ->
        printf("Bank Process: bank verifying
        password...\n");
        if
::      bank2consortium!VERIFIED_PW;
        printf("Bank Process: bank verified acc...\n");
        goto ProcessTransaction;
::      bank2consortium!WRONG_PW;
        printf("Bank Process: wrong password,
        authentication fails...\n");
        goto VerifyPassword;
        fi;
::      consortium2bank?END_TRANSACTION -> goto end;
        fi;
        ProcessTransaction:
        if
:: consortium2bank?CONSORTIUM_PROCESS_TRANS ->
        printf("Bank Process: bank process
        transaction...\n");
        if
::      bank2consortium!BANK_TRANS_SUCCESS;
        printf("Bank Process: bank transaction
        succesful...\n");
::      bank2consortium!BANK_TRANS_FAIL;
        printf("Bank Process: bank transaction
        fail...\n");
        fi;
:: consortium2bank?END_TRANSACTION -> goto end;      fi;
        if
:: consortium2bank?ANOTHER_TRANSACTION -> goto
ProcessTransaction;
:: consortium2bank?END_TRANSACTION -> goto end;      fi;
        end:
        printf("Bank Process: ending ...\n");
        }
```

Gambar 4.13 Definisi *proctype* Bank

4.4.4 Definisi *proctype User*

User merupakan sebuah entitas yang memiliki interaksi dengan ATM. User memiliki peran memasukan kartu ke dalam mesin ATM, Memasukan password, memilih menu pada mesin ATM dan proses pembatalan transaksi.

Pada proses memasukan kartu ke dalam mesin, user memiliki 2 *state,* yaitu user memasukan kartunya ke dalam mesin ATM, kemudian ATM akan melakukan pemeriksaan mengenai validitas dari kartu tersebut. Jika kartu tersebut valid, maka akan dilanjutkan untuk memasukan password. Kedua, jika kartu yang dimasukan itu tidak valid, maka mesin ATM akan mengeluarkan kartu.

Pada proses memasukan password, user memiliki 4 state, yaitu user melakukan permohonan kepada ATM untuk memasukan password, maka ATM akan mengizinkan user untuk memasukan password. Kedua, setelah user memasukan password, maka ATM akan memeriksa, jika password yang dimasukan adalah benar, maka user akan diteruskan untuk memilih menu yang ada pada mesin ATM. Ketiga, jika password yang dimasukan oleh user adalah salah, maka kartu akan dikeluarkan. Terakhir, jika user membatalkan untuk memasukan password, maka kartu juga akan dikeluarkan.

Pada proses memilih menu pada mesin ATM, user memiliki 9 *state,* yaitu user melakukan permohonan kepada ATM untuk melakukan transaksi, kemudian jika user memilih untuk melakukan penarikan uang, maka ATM akan mengarahkan user ke dalam menu penarikan uang. Kedua, setelah user masuk ke dalam menu penarikan uang, maka user diminta untuk memasukan jumlah uang yang akan diambil. Ketiga, setelah transaksi berhasil, maka user dapat mengambil uang. Keempat, jika transaksi gagal, maka user akan dilanjutkan ke dalam proses pembatalan. Kelima, akan ditampilkan pemberitahuan bahwa user telah membatalkan transaksi penarikan uang tersebut, dilanjutkan dengan dikeluarkannya kartu. Keenam, user dapat

memilih menu untuk melihat saldo. Ketujuh, jika transaksi tersebut berhasil maka mesin ATM akan menampilkan saldo user, kemudian akan dilanjutkan kedalam proses pembatalan. Kedelapan, jika transaksi tersebut tidak berhasil, maka akan dilanjutkan ke dalam proses pembatalan. Terakhir, jika user melakukan pembatalan transaksi, maka kartu akan dikeluarkan.

Pada proses pembatalan (*terminate)* transaksi, user memiliki 2 *state*, yaitu jika user tidak meneruskan transaksi lain, maka kartu akan dikeluarkan. Kedua, jika user memilih untuk meneruskan ke transaksi lain, maka user akan masuk ke dalam menu pilihan untuk memilih menu selanjutnya.

Empat proses tersebut didefinisikan ke dalam bahasa PROMELA dalam sebuah rangkaian *if statement* seperti pada Gambar 4.14.

```
        proctype User(){
        printf("initiating User Process...\n");
        InsertCard:
        atm2user?DISPLAY_MAIN_MENU;
        printf("User Process: user inserting
        card...\n");
        user2atm!INSERT_CARD;
        if
::      atm2user?VALID_CARD ->
        goto EnterPassword;
::      atm2user?INVALID_CARD ->
        goto RemoveCard;
        fi;
        EnterPassword:
        atm2user?REQUEST_PASSWORD;
        if
::      printf("User Process: user entered
        password...\n");
        user2atm!ENTERED_PASSWORD;
        if
::      atm2user?CORRECT_PW ->
        goto EnterSelection;
```

```
::      atm2user?INCORRECT_PW ->
        goto RemoveCard;
        fi;
::      printf("User Process: user cancel
        transactions...\n");
        user2atm!CANCEL;
        goto RemoveCard;
        fi;
        EnterSelection:
        atm2user?REQ_TRANSACTION;z
        if
::      printf("User Process: user select
        withdrawal...\n");
        user2atm!WITHDRAW;
        atm2user?REQ_AMT;
        if
::      printf("User Process: user enter withdrawal
        amt...\n");
        user2atm!ENTERED_AMT;
        if
::      atm2user?TRANSACTION_SUCCESS ->
        printf("User Process: user take cash...\n");
        user2atm!TOOK_CASH;
        goto Terminate;
::      atm2user?TRANSACTION_UNSUCCESS ->
        goto Terminate;
        fi;
::      printf("User Process: user cancel
        transactions...\n");
        user2atm!CANCEL;
        goto RemoveCard;
        fi;
::      printf("User Process: user select
        enquiry...\n");
        user2atm!ENQUIRY;
        if
::      atm2user?TRANSACTION_SUCCESS ->
        printf("User Process: user view amount in
        bank...\n");
        goto Terminate;
::      atm2user?TRANSACTION_UNSUCCESS ->
        goto Terminate;
```

```
        fi;
::      printf("User Process: user cancel
        transactions...\n");
        user2atm!CANCEL;
        goto RemoveCard;
        fi;
if
::      printf("User Process: user select
        transfer...\n");
        user2atm!TRANSFER;
        atm2user?REQ_ACC_NO;
        if
::      printf("User Process: user choose bank
        number...\n");
        user2atm!ENTERED_NO;
        if
::      atm2user?TRANSACTION_SUCCESS ->
        printf("User Process: transfer
        successfully...\n");
        goto Terminate;
::      atm2user?TRANSACTION_UNSUCCESS ->
        goto Terminate;
        fi;
::      printf("User Process: user cancel
        transactions...\n");
        user2atm!CANCEL;
        goto RemoveCard;
fi;
        Terminate:
        atm2user?REQ_CONTINUE;
        if
::      printf("User Process: user choose not continue
        with another transaction...\n");
        user2atm!TERMINATE;
        goto RemoveCard;
::      printf("User Process: user choose to continue
        with another transaction...\n");
        user2atm!CONTINUE;
        goto EnterSelection;
        fi;
        RemoveCard:
        atm2user?REQ_REMOVE_CARD ->
        printf("User Process: user remove card...\n");
        user2atm!REMOVED_CARD;
}
```

Gambar 4. 14 Definisi *proctype* User

4.4.5 Definisi *proctype ATM*

ATM merupakan sebuah entitas yang memiliki interaksi dengan user dan konsorsium. ATM memiliki peran membaca kartu, menerima akun, memverifikasi akun, permohonan untuk melanjutkan transaksi, permohonan untuk melakukan masukan jumlah uang yang akan ditarik, transaksi penarikan uang, transaksi menampilkan saldo user, transaksi transfer sesama bank, mengeluarkan uang, mencetak tanda bukti dan mengeluarkan kartu.

Pada proses membaca kartu ATM memiliki 2 *state,* yaitu jika kartu dapat terbaca oleh ATM, maka akan dilanjutkan ke proses menerima akun. Kedua, jika tidak maka kartu akan dikeluarkan.

Pada proses menerima akun ATM memiliki 2 *state*, yaitu jika user memasukan password, maka ATM akan melakukan veriikasi password bersama dengan konsorsium. ATM akan melakukan komunikasi dengan konsorsium untuk memastika akun tersebut dapat diterima atau tidak. Setelah itu, ATM akan masuk ke dalam proses verifikasi akun (VerifyAccState). Kedua, jika user melakukan pembatalan transaksi pada mesin ATM, maka ATM akan memproses pengeluaran kartu, sehingga pada akhirnya kartu akan dikeluarkan.

Pada proses memverifikasi akun, ATM memiliki 2 *state*, yaitu jika konsorsium memberikan informasi bahwa password yang telah dimasukan oleh user adalah benar, maka ATM melanjutkan ke proses permohonan transaksi (ReqTransactionState). Kedua, jika konsorsium memberikan informasi bahwa password yang telah dimasukan oleh user adalah salah, maka ATM melanjutkan ke dalam proses mencetak bukti (PrintReceiptState).

Pada proses permohonan untuk melanjutkan transaksi, ATM memiliki 3 *state,* yaitu jika user melakukan pembatalan transaksi pada mesin ATM, maka ATM akan mengeluarkan kartu. Kedua, jika user memilih untuk melakukan penarikan uang

tunai, maka mesin ATM akan mengarahkan ke menu penarikan uang tunai dan akan meminta user untuk memasukan jumlah uang tunai yang ingin ditarik. Ketiga, jika user memilih untuk melakukan pemeriksaan saldo, maka ATM akan mengarahkan user ke menu pemeriksaan saldo.

Pada proses permohonan untuk melakukan masukan jumlah uang yang akan ditarik, ATM memiliki 2 *state,* yaitu jika user melakukan pembatalan transaksi pada mesin ATM, maka ATM akan mengeluarkan kartu. Kedua, jika user memasukan jumlah uang yang akan ditarik pada mesin ATM, maka ATM akan melanjutkan ke proses transaksi penarikan uang tunai (WithdrawalTransactionState).

Pada proses penarikan uang tunai, ATM memiliki 2 *state,* yaitu jika konsorsium menyatakan bahwa transaksi berhasil, maka ATM akan melanjutkan ke proses pengeluaran uang tunai (CashDispenseState). Kedua, jika konsorsium menyatakan transaksi tidak berhasil, maka ATM akan melanjutkan ke transaksi ReqContinueState.

Pada transaksi menampilkan saldo, ATM memiliki 2 *state,* yaitu jika konsorsium menyatakan transaksi berhasil, maka ATM akan melanjutkan ke proses ReqContinueState dengan parameter transaksi berhasil. Kedua, jika konsorsium menyatakan transaksi tidak berhasil, maka ATM akan melanjutkan ke proses ReqContinueState dengan parameter transaksi gagal.

Pada transaksi transfer uang sesama Bank, ATM memiliki 2 *state*, yaitu jika konsorsium menyatakan transaksi berhasil, maka ATM akan melanjutkan ke proses transfer (TransferTransactionState). Kedua, jika konsorsium menyatakan transaksi tidak berhasil, maka ATM akan melanjutkan ke transaksi ReqContinueState.

Pada proses mengeluarkan uang tunai, ATM harus memastikan bahwa password benar, user tidak menekan tombol batal, menu penarikan uang tunai dipilih oleh user, user tidak memilih menu pemeriksaan saldo, transaksi berhasil dan jumlah

uang tunai telah dimasukan. Setelah, ATM mengeluarkan uang tunai, maka ATM akan melanjutkan ke proses ReqContinueState.

Pada proses mencetak bukti, ATM harus memastikan bahwa user tidak mengeluarkan kartu, setelah bukti dicetak, maka ATM akan melanjutkan ke proses mengeluarkan kartu (EjectCardState).

Pada proses mengeluarkan kartu, ATM akan mengeluarkan kartu dari mesin ATM, sehingga segala transaksi akan berakhir. Sepuluh proses tersebut didefinisikan ke dalam bahasa PROMELA dalam sebuah rangkaian *if statement* seperti pada Gambar 4.15.

```
proctype ATM(){
       printf("initiating ATM Process...\n");
       DisplayMenuState:
       atm2user!DISPLAY_MAIN_MENU ->
       printf("ATM Process: Main Menu...\n");
       user2atm?INSERT_CARD ->
       goto
ReadCardState;
ReadCardState:
       if
::     printf("ATM Process: Card is valid...\n");
       atm2user!VALID_CARD;
       printf("ATM Process: Prompt user for
       password...\n");
       atm2user!REQUEST_PASSWORD;
       goto ReceiveAccState;
::     printf("ATM Process: Card is
       unreadable...\n");
       atm2user!INVALID_CARD;
       atm2user!REQ_REMOVE_CARD;
       goto EjectCardState;
       fi;
ReceiveAccState:
       if
:: user2atm?ENTERED_PASSWORD ->
       printf("ATM Process: ATM verifying password
       with consortium...\n");
       atm2consortium!VERIFY_PW_CON;
       user_cancel = false;
       goto VerifyAccState;
:: user2atm?CANCEL ->
       printf("ATM Process: User cancel
       transaction. Eject card. ...\n");
       atm2user!REQ_REMOVE_CARD;
       user_cancel = true;
       goto EjectCardState;
       fi;
VerifyAccState:
       if
:: consortium2atm?CORRECT_PW ->
       atm2user!CORRECT_PW;
```

```
       printf("ATM Process: password is verified
       by consortium\n ");
       password_correct = true;
       atm2user!REQ_TRANSACTION;
       printf("ATM Process: Prompt user to enter
       in the transaction type\n ");
       goto ReqTransactionState;
:: consortium2atm?INCORRECT_PW   ->
       printf("ATM Process: password is
       incorrect\n");
       atm2user!INCORRECT_PW;
       password_correct = false;
       goto PrintReceiptState;
       fi;
ReqTransactionState:
       if
:: user2atm?CANCEL ->
printf("ATM Process: User cancel transaction.
Eject card. ...\n");
       atm2user!REQ_REMOVE_CARD;
       user_cancel = true;
       goto EjectCardState;
:: user2atm?WITHDRAW ->
       printf("ATM Process: Prompt user for
withdrawal amt...\n");
atm2user!REQ_AMT;
       withdraw_selected = true;
       enquiry_selected = false;
       user_cancel = false;
       goto ReqAmtState;
:: user2atm?TRANSFER ->
       printf("ATM Process: Prompt user for
transfer...\n");
atm2consortium!ATM_PROCESS_TRANS;
       withdraw_selected = false;
       enquiry_selected = false;
       transfer_selected = true;
       user_cancel = false;
       goto TransferTransactionState;
:: user2atm?ENQUIRY ->
       printf("ATM Process: Prompt user for
withdrawal amt...\n");
```

```
atm2consortium!ATM_PROCESS_TRANS;
       withdraw_selected = false;
       enquiry_selected = true;
       user_cancel = false;
       goto EnquiryTransactionState;
       fi;

ReqAmtState:
       if
:: user2atm?CANCEL ->
       printf("ATM Process: User cancel
transaction. Eject card. ...\n");
       atm2user!REQ_REMOVE_CARD;
       user_cancel = true;
       amount_entered = false;
       goto EjectCardState;
:: user2atm?ENTERED_AMT ->
       printf("ATM Process: process withdrawal
transaction...\n");
       atm2consortium!ATM_PROCESS_TRANS;
       user_cancel = false;
       amount_entered = true;
       goto WithdrawalTransactionState;
:: user2atm?ENTERED_AMT_TRF ->
       printf("ATM Process: process transfer
transaction...\n");
       atm2consortium!ATM_PROCESS_TRANS;
       user_cancel = false;
       amount_entered = true;
       goto TransferTransactionState;
       fi;
WithdrawalTransactionState:
       if
:: consortium2atm?TRANSACTION_SUCCESS   ->
printf("ATM Process: Dispense cash. Prompt user
to take cash...\n");
       atm2user!TRANSACTION_SUCCESS;
       transaction_success = true;
       goto CashDispenseState;
:: consortium2atm?TRANSACTION_UNSUCCESS ->
       printf("ATM Process: Bank transaction
       failed\n");
       atm2user!TRANSACTION_UNSUCCESS;
```

```
       atm2user!REQ_CONTINUE;
       transaction_success = false;
       goto ReqContinueState;
       fi;
       EnquiryTransactionState:
       if
:: consortium2atm?TRANSACTION_SUCCESS   ->
       printf("ATM Process: Display amt to
       user...\n");
       atm2user!TRANSACTION_SUCCESS;
       atm2user!REQ_CONTINUE;
       transaction_success = true;
       goto ReqContinueState;

:: consortium2atm?TRANSACTION_UNSUCCESS ->
       printf("ATM Process: Bank transaction
       failed\n");
       atm2user!TRANSACTION_UNSUCCESS;
       atm2user!REQ_CONTINUE;
       transaction_success = false;
       goto ReqContinueState;
       fi;
TransferTransactionState:
       if
:: consortium2atm?TRANSACTION_SUCCESS   ->
printf("ATM Process: Transfer cash. Prompt user
to transfer...\n");
       atm2user!TRANSACTION_SUCCESS;
       transaction_success = true;
       goto ReqAmtState;
:: consortium2atm?TRANSACTION_UNSUCCESS ->
printf("ATM Process: Bank transaction failed\n");
atm2user!TRANSACTION_UNSUCCESS;
atm2user!REQ_CONTINUE;
       transaction_success = false;
       goto ReqContinueState;
       fi;
CashDispenseState:
       user2atm?TOOK_CASH ->
       printf("ATM Process: Prompt user to
       continue with another transaction or
       terminate the transaction...\n");
       atm2user!REQ_CONTINUE;
```

```
assert(password_correct);
       assert(!user_cancel);
       assert(withdraw_selected);
       assert(!enquiry_selected);
       assert(transaction_success);
       assert(amount_entered);
       cash_dispensed = true;
       goto ReqContinueState;
       ReqContinueState:
/*
/* to model single transaction mode *
continue_transaction = false;
       goto PrintReceiptState;
*/

/* to model continuous transaction mode  */
       If
:: user2atm?TERMINATE ->
       continue_transaction = false;
       goto PrintReceiptState;
:: user2atm?CONTINUE ->
       printf("ATM Process: Redirect user to
       select transaction type...\n");
       atm2user!REQ_TRANSACTION;
       atm2consortium!ANOTHER_TRANSACTION;
       continue_transaction = true;
       goto ReqTransactionState;
       fi;
/**/

PrintReceiptState:
       printf("ATM Process: print receipt ...\n");
       atm2user!REQ_REMOVE_CARD;
       receipt_printed = true;
       goto EjectCardState;
EjectCardState:
       printf("ATM Process: eject card...\n");
       card_ejected = true;
       user2atm?REMOVED_CARD;
       atm2consortium!END_TRANSACTION;
       }
```

Gambar 4.15 Definisi *proctype* ATM

4.5 Pemeriksaan Properti

Untuk lingkup tugas akhir ini, berusaha untuk memeriksa sifat dari transaksi penarikan yang berhasil. Sebuah transaksi penarikan yang berhasil adalah transaksi di mana pengguna memilih opsi penarikan dan transaksi penarikan berhasil dan ATM mengeluarkan uang tunai pada akhirnya. Dalam hal ini, akan dilakukan Pemeriksaan dalam dua tahap: pemeriksaan awal untuk menyatakan transaksi penarikan uang dan pemeriksaan akhir untuk sifat *liveness* dan *safety* dari transaksi penarikan uang.

4.5.1 Pemeriksaan Awal

Untuk mulai memeriksa properti, pertama kita menempatkan *Booleans* dalam kode ATM Promela sebagai *checkpoint* dalam proses ATM mana saja sifat yang harus diperiksa. *Booleans* tidak ditempatkan dalam proses Bank dan Konsorsium karena kegagalan transaksi pada titik-titik ini akan menjadi *feedback* untuk proses ATM. Proses pengguna tidak memiliki *Booleans* karena manusia adalah pihak eksternal dan perilaku manusia berada di luar kendali dari mesin ATM. Di bawah ini adalah *Booleans* yang diyatakan dan ditempatkan dalam kode ATM Promela.

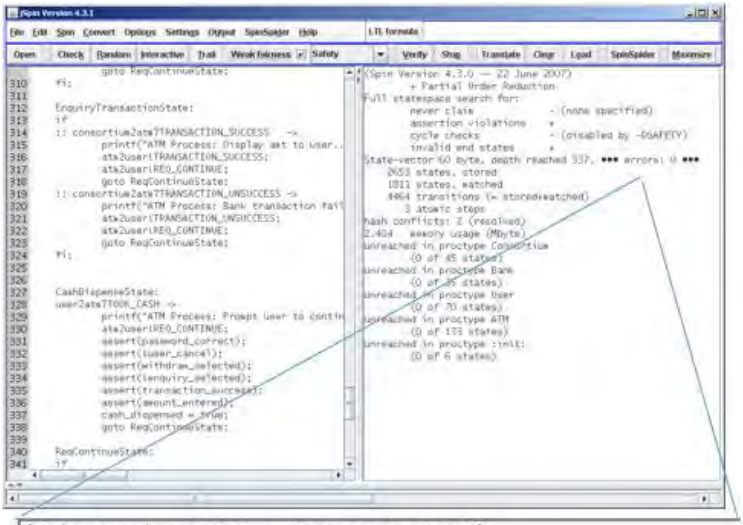
Ketika *checkpoint* ditambahkan, kita akan melakukan pemeriksaan cepat dengan menempatkan pernyataan di CashDispenseState dari proses ATM seperti pada Gambar 4.16. Pernyataan akan menegaskan sifat berikut ketika uang tunai tidak ada.

1. **password_correct**: pin benar dimasukan oleh pengguna

2. **!user_cancel**: Pengguna tidak harus membatalkan pada setiap titik transaksi
3. **withdraw_selected**: Pengguna harus memilih menu penarikan uang selama pemilihan transaksi
4. **!enquiry_selected**: Pengguna tidak harus memilih cek saldo selama seleksi transaksi
5. **transaction_success**: transaksi penarikan berhasil
6. **amount_entered**: pengguna memasukan jumlah uang yang akan ditarik

Pada pemeriksaan awal ini didefinisikan 2 properti, yaitu properti 1 dan properti 2. Pemeriksaan awal ini ditujukan untuk mengetahui jika transaksi memasuki *exception condition* atau sebuah kondisi pengecualian atau dapat dikatakan sebuah kondisi yang tidak diinginkan, maka apakah mesin ATM akan mengeluarkan kartu. Karena dengan mesin ATM mengeluarkan kartu, transaksi akan berakhir dan kondisi yang tidak diinginkan tidak akan terjadi.

Setelah dilakukan pemeriksaan dengan menggunakan SPIN diperoleh hasil pada Gambar 4.16 dan 4.18. Dari hasil tersebut tidak ditemukan galat atau *eror*.



```
(Spin Version 6.0.0 -- 5 December 2010)
        + Partial Order Reduction
Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        cycle checks            - (disabled by -DSAFETY)
        invalid end states      +
State-vector 60 byte, depth reached 414, *** errors: 0 ***
     3206 states, stored
     2319 states, matched
     5525 transitions (= stored+matched)
        3 atomic steps
hash conflicts:         6 (resolved)
    2.391       memory usage (Mbyte)
unreached in proctype Consortium
        (0 of 45 states)
unreached in proctype Bank
        (0 of 35 states)
unreached in proctype User
        (0 of 70 states)
unreached in proctype ATM
        (0 of 135 states)
unreached in init
        (0 of 6 states)
pan: elapsed time 0.009 seconds
```

Gambar 4.16 Hasil Ekesekusi Program

Properti 1	Jika transaksi penarikan dibatalkan dan pengguna memilih untuk tidak melanjutkan transaksi, maka ATM akhirnya akan mencetak tanda dan mengeluarkan kartu.
Properti LTL	G (cash_dispensed Λ ¬continue_transaction → F (receipt_printed Λ card_ejected))
Sintaks SPIN	`[](cash_dispensed && !continue_transaction -> <>(receipt_printed && card_ejected))`

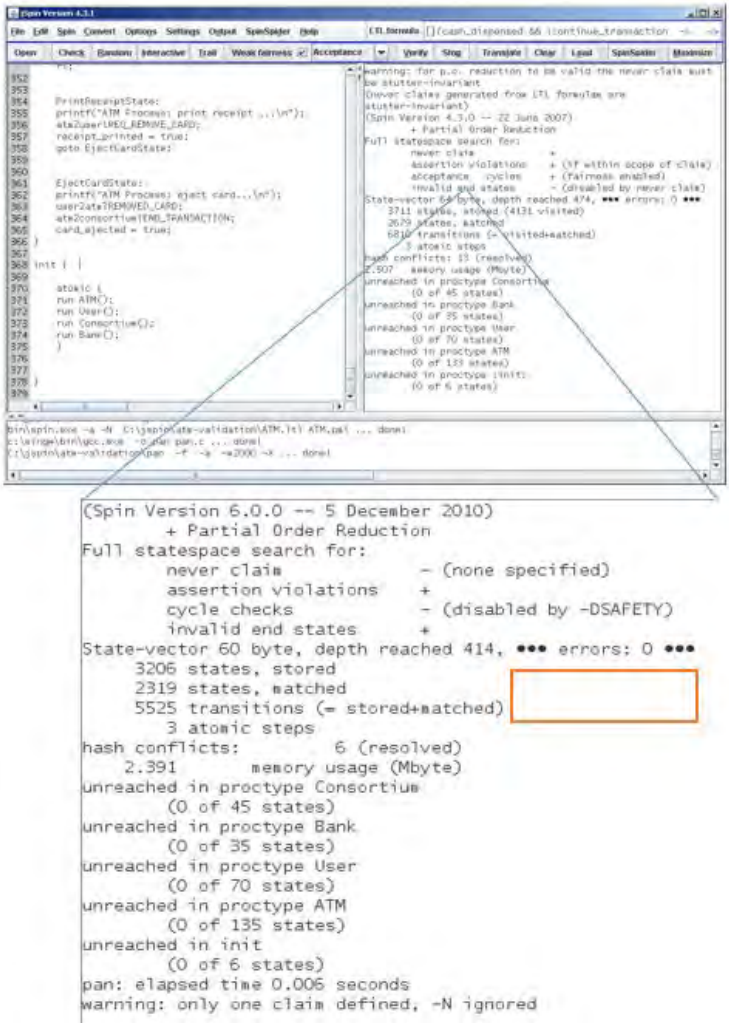
Büchi automata	`never { /* !( [](receipt_printed -> <>card_ejected) ) */`
	`T0_init:`
	`if`
	`:: (! ((card_ejected)) && (receipt_printed)) -> goto accept_S4`
	`:: (1) -> goto T0_init`
	`fi;`
	`accept_S4:`
	`if`
	`:: (! ((card_ejected))) -> goto accept_S4`
	`fi;`
	`}`

Properti 2	Jika ATM mencetak tanda terima, maka akan mengeluarkan kartu.
Properti LTL	G (receipt_printed → F card_ejected)

Sintaks SPIN	`[](receipt_printed -> <>card_ejected)`

Büchi automata	`never { /* !([](cash_dispensed && !continue_transaction -> <>(receipt_printed && card_ejected))) */`
	`T0_init:`
	`if`
	`:: (((! ((card_ejected)) && ! ((continue_transaction)) && (cash_dispensed)) || (! ((continue_transaction)) && ! ((receipt_printed)) && (cash_dispensed)))) -> goto accept_S3`
	`:: (1) -> goto T0_init`
	`fi;`
	`accept_S3:`
	`if`
	`:: (((! ((card_ejected))) || (! ((receipt_printed))))) -> goto accept_S3`
	`fi;`
	`}`

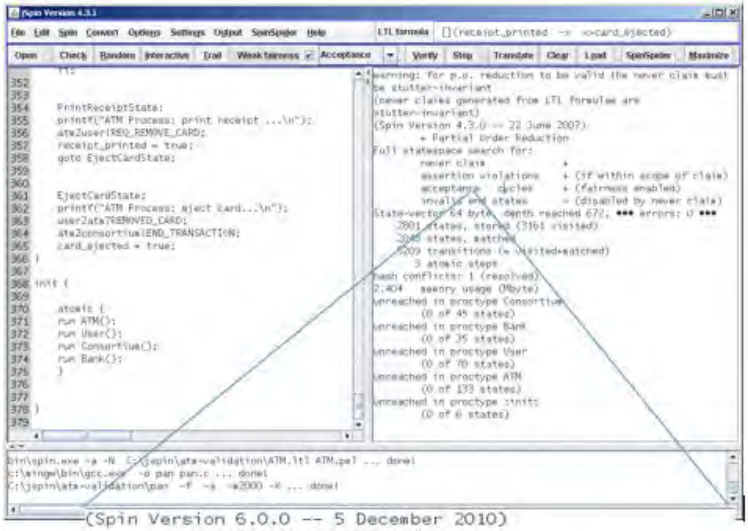
Properti 1 : `[](cash_dispensed && !continue_transaction -> <>(receipt_printed && card_ejected))`



```
(Spin Version 6.0.0 -- 5 December 2010)
        + Partial Order Reduction
Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        cycle checks            - (disabled by -DSAFETY)
        invalid end states      +
State-vector 60 byte, depth reached 414, ••• errors: 0 •••
     3206 states, stored
     2319 states, matched
     5525 transitions (= stored+matched)
        3 atomic steps
hash conflicts:         6 (resolved)
    2.391       memory usage (Mbyte)
unreached in proctype Consortium
        (0 of 45 states)
unreached in proctype Bank
        (0 of 35 states)
unreached in proctype User
        (0 of 70 states)
unreached in proctype ATM
        (0 of 135 states)
unreached in init
        (0 of 6 states)
pan: elapsed time 0.006 seconds
warning: only one claim defined, -N ignored
```

Gambar 4.17 Hasil Eksekusi untuk Properti 1

Properti 2 : `[](receipt_printed -> <>card_ejected)`



```
(Spin Version 6.0.0 -- 5 December 2010)
        + Partial Order Reduction
Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        cycle checks            - (disabled by -DSAFETY)
        invalid end states      +
State-vector 60 byte, depth reached 414, ••• errors: 0 •••
     3206 states, stored
     2319 states, matched
     5525 transitions (= stored+matched)
        3 atomic steps
hash conflicts:         6 (resolved)
    2.391       memory usage (Mbyte)
unreached in proctype Consortium
        (0 of 45 states)
unreached in proctype Bank
        (0 of 35 states)
unreached in proctype User
        (0 of 70 states)
unreached in proctype ATM
        (0 of 135 states)
unreached in init
        (0 of 6 states)
pan: elapsed time 0.006 seconds
warning: only one claim defined, -N ignored
```

Gambar 4.18 Hasil Eksekusi untuk Properti 2

4.5.2 PEMERIKSAAN PROPERTI *LIVENESS* DAN *SAFETY*

Pada pemeriksaan tahap 2 ini adalah akan diperiksa properti *Liveness* dan *Safety*. Pemeriksaan properti – properti tersebut ditujukan agar dapat diketahui bahwa segala fungsi vital dari sistem ATM tersebut berjalan dengan baik atau tidak. Dari properti ini dapat diketahui mengenai celah keamanan yang mungkin terjadi.

Pada pemeriksaan properti *Liveness,* didefinisikan 3 properti yaitu *Liveness 1, Liveness 2* dan *Liveness 3*. Pada pemeriksaan properti *Liveness*, pemeriksaan berfokus pada transaksi pengambilan uang tunai. Hasil pemeriksaan ditampilkan pada Gambar 4.20.

Pada pemeriksaan properti *Safety,* didefinisikan 4 properti yaitu *Safety 1, Safety 2, Safety 3* dan *Safety 4.* Pada pemeriksaan properti *Safety,* pemeriksaan berfokus pemeriksaan akun dan proses permintaan transaksi yang lain. Hasil pemeriksaan ditampilkan pada Gambar 4.21.

Liveness 1	Jika pengguna memilih penarikan dan transaksi berhasil, maka ATM akhirnya akan mengeluarkan uang tunai.
Properti LTL	G (withdraw_selected Λ transaction_success → F cash_dispensed)
Sintaks SPIN	`[](withdraw_selected && transaction_success -> <> cash_dispensed)`
Liveness 2	Jika ATM mengeluarkan uang tunai, maka akan mencetak tanda terima.

Properti LTL	G (cash_dispensed → F receipt_printed)
Sintaks SPIN	`[](cash_dispensed -> <> receipt_printed)`
Liveness 3	Jika ATM mengeluarkan uang tunai, maka akan mengeluarkan kartu.
Properti LTL	G (cash_dispensed → F card_ejected)
Sintaks SPIN	`[](cash_dispensed -> <> card_ejected)`
Safety 1	Jika pengguna memasukkan pin yang salah, maka ATM tidak akan mengeluarkan uang tunai.
Properti LTL	G ¬ (¬password_correct Λ cash_dispensed)
Sintaks SPIN	`[]!(!password_correct && cash_dispensed)`
Safety 2	Jika transaksi di ATM gagal, maka ATM tidak akan mengeluarkan uang tunai.

Properti LTL	G ¬ (¬transaction_success Λ cash_dispensed)
Sintaks SPIN	`[]!(!transaction_success && cash_dispensed)`
Safety 3	Jika pengguna tidak memilih untuk melakukan transaksi penarikan, maka ATM tidak akan mengeluarkan uang tunai.
Properti LTL	G ¬ (¬withdraw_selected Λ cash_dispensed)
Sintaks SPIN	`[]!(!withdraw_selected && cash_dispensed)`
Safety 4	Jika pengguna memilih untuk melakukan transaksi cek saldo, maka ATM tidak akan mengeluarkan uang tunai.
Properti LTL	G ¬(enquiry_selected Λ cash_dispensed)
Sintaks SPIN	`[]!(enquiry_selected && cash_dispensed)`

Akan diperiksa untuk keberhasilan dalam transaksi penarikan tunggal, di mana pengguna melakukan transaksi penarikan dan mengakhiri transaksi setelah kas ditiadakan; pengguna tidak meneruskan transaksi lainnya setelah penarikan. Untuk model

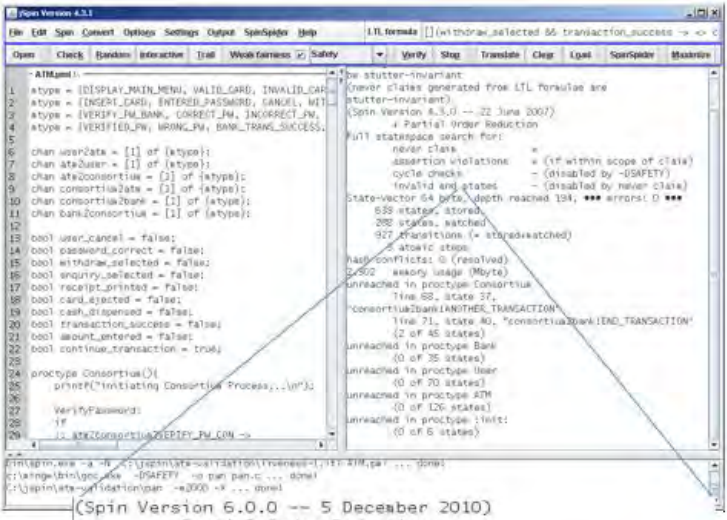
modus transaksi ini, telah diberikan komentar untuk modus transaksi terus menerus atau kontinu dan telah dilakukan *uncomment* pada proses ReqContinueState ATM, seperti pada Gambar 4.19.

```
        ReqContinueState:

        /* to model single transaction mode */
        continue_transaction = false;
        goto PrintReceiptState;

/*
        /* to model continuous transaction mode  *
        if
        :: user2atm?TERMINATE ->
            continue_transaction = false;
            goto PrintReceiptState;
        :: user2atm?CONTINUE ->
            printf("ATM Process: Redirect user to
select transaction type...\n");
            atm2user!REQ_TRANSACTION;
            atm2consortium!ANOTHER_TRANSACTION;
            continue_transaction = true;
            goto ReqTransactionState;
        fi;
*/
```

Gambar 4. 19 proses *ReqContinueState* ATM

Untuk memverifikasi property LTL di atas, telah dimuat berkas property LTL dan telah disematkan pada SPIN, kemudian diterjemahkan kedalam Büchi automata dan dijalankan verifikasi dalam *Acceptance Mode.* SPIN memberikan laporan bahwa tidak ada kesalahan untuk semua property LTL tersebut. Hasil verifikasi untuk *liveness 1* dan *safety 1* ditampilkan pada Gambar 4.20 – 4.21.

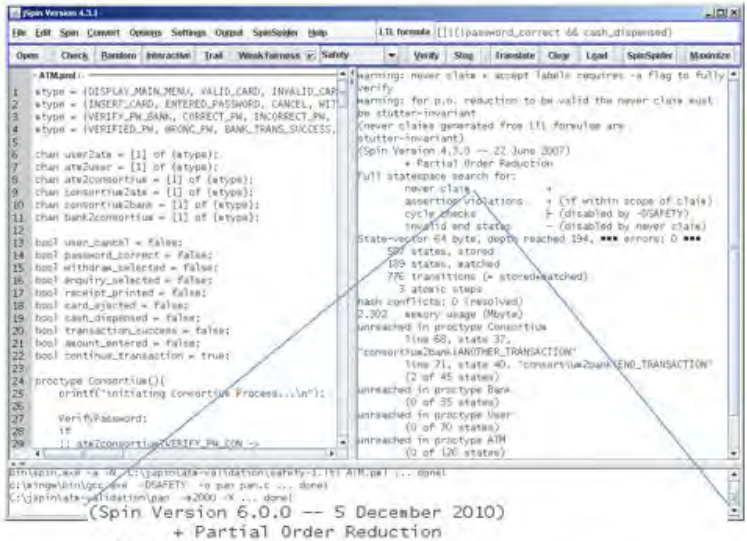
Liveness 1: `[](withdraw_selected && transaction_success -> <> cash_dispensed)`



```
(Spin Version 6.0.0 -- 5 December 2010)
        + Partial Order Reduction
Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        cycle checks            - (disabled by -DSAFETY)
        invalid end states      +
State-vector 60 byte, depth reached 414, ••• errors: 0 •••
     3206 states, stored
     2319 states, matched
     5525 transitions (= stored+matched)
        3 atomic steps
hash conflicts:         6 (resolved)
    2.391       memory usage (Mbyte)
unreached in proctype Consortium
        (0 of 45 states)
unreached in proctype Bank
        (0 of 35 states)
unreached in proctype User
        (0 of 70 states)
unreached in proctype ATM
        (0 of 135 states)
unreached in init
        (0 of 6 states)
pan: elapsed time 0.006 seconds
warning: only one claim defined, -N ignored
```

Gambar 4. 20 Hasil Eksekusi untuk Liveness 1

Safety 1: `[]!(!password_correct && cash_dispensed)`



```
(Spin Version 6.0.0 -- 5 December 2010)
        + Partial Order Reduction
Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        cycle checks            - (disabled by -DSAFETY)
        invalid end states      +
State-vector 60 byte, depth reached 414, *** errors: 0 ***
     3206 states, stored
     2319 states, matched
     5525 transitions (= stored+matched)
        3 atomic steps
hash conflicts:         6 (resolved)
    2.391       memory usage (Mbyte)
unreached in proctype Consortium
        (0 of 45 states)
unreached in proctype Bank
        (0 of 35 states)
unreached in proctype User
        (0 of 70 states)
unreached in proctype ATM
        (0 of 135 states)
unreached in init
        (0 of 6 states)
pan: elapsed time 0.008 seconds
warning: only one claim defined, -N ignored
```

Gambar 4.21 Hasil Eksekusi untuk Safety 1

BAB V

PENUTUP

5.1 Kesimpulan

Dalam tugas akhir ini telah berhasil memodelkan sistem ATM dalam bahasa PROMELA dan memverifikasi model tersebut dengan *tools* model pemeriksaan SPIN. Dalam penelitian ini telah ditunjukan proses verifikasi dalam dua tahap. Pada tahap pertama, kode telah diverifikasi dan tidak ditemukan kode yang *deadlock* dan *unreachable*. Untuk proses transaksi dalam sistem tersebut juga telah dilakukan Pemeriksaan terhadap dua properti LTL dan eror tidak ditemukan. Pada tahap 2, dilakukan pemeriksaan terhadap property *Liveness* dan *Safety*, telah dikonstruksi enam formula LTL untuk memverifikasi transaksi dalam model ATM tersebut. Pada pemeriksaan tahap 2, menunjukan bahwa model ATM telah lulus uji properti *Liveness* dan *Safety*.

Pada kesimpulan akhir dalam tugas akhir ini menyatakan bahwa model sistem ATM tersebut aman karena tidak ditemukan kode yang *deadlock* dan *unreachable* serta sistem tersebut layak untuk digunakan karena telah lulus uji *Liveness* dan *Safety.*

5.2 Saran

Pada penelitian selanjutnya, direkomendasikan untuk melakukan beberapa hal berikut ini :

1. Untuk model ATM diasumsikan akan menelan kartu jika telah salah memasukan PIN sebanyak lebih dari tiga kali.
2. Diberikan model waktu tunggu sebesar lima detik dalam hal transaksi yang gagal
3. Diberikan model pengalihan fungsi transaksi dana jika terjadi kesalahan transfer.

DAFTAR PUSTAKA

[1] H. Shi, W. Ma, M. Yang and X. Zhang, "A Case Study of Model Checking Retail Banking," *Journal of Computers,* vol. 7, pp. 2503-2510, October 2012.

[2] Y. Wang, Y. Zhang, X. Li and H. Guo, "The Formal Design Model of an Automatic Teller Machine (ATM)," *International Journal of Software Science and Computational Intelligence,* vol. 2, pp. 102-131, 2010.

[3] M. d. M. Gallardo, J. Mart´ınez, P. Merino and E. Pimentel, "A Tool for Abstraction in Model Checking," *Electronic Notes in Theoretical Computer Science,* vol. 66, pp. 17-32, 2002.

[4] X. Yu, Z. Wang, G. Pu, D. Mao and J. Liu, "The Verification of rCOS Using Spin," *Electronic Notes in Theoretical Computer Science,* vol. 207, pp. 49-67, 2008.

[5] H. Groefsema and D. Bucur, "A Survey of Formal Business Process Verification: From Soundness to Variability," University of Groningen, Groningen.

[6] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorensen, Object Oriented Modeling and Design, New Jersey: Prentice-Hall Inc., 1991.

[7] C. Baier and J. P. Katoen, Principles of Model Checking, Massachusetts: The MIT Press.

[8] G. Holzmann, The SPIN Model Checker: Primer and Reference Manual, Addison Wesley, 2004.

[9] K. Havelund, J. Penix and W. Visser, SPIN Model Checking and Software Verification, California: Springer, 2006.

[10] J. D. Franklin, Abstractions for Model Checking System Security, Pittsburgh: Jason Douglas Franklin, 2012.

[11] J. Tretmans, "Model Based Testing with Labelled Transition Systems," Embedded Systems Institute, Eindhoven, 2012.

[12] G. J. Holzmann, "The Model Checker SPIN," *IEEE Transactions On Software Engineering,* vol. 23, 1997.

BIODATA PENULIS

Penulis memiliki nama lengkap Ikhwan Mohamamd Iqbal atau biasa dipanggil Ikhwan, lahir di Jakarta, 5 Juni 1994. Terlahir sebagai anak pertama dari 3 bersaudara.

Sejak usia enam tahun, penulis mulai bersekolah formal di SDN Lagoa 01 Jakarta (2000-2006), SMPN 279 Jakarta (2006-2009), SMAN 52 Jakrta (2009-2012). Setelah lulus SMAN, penulis melanjutkan studi ke jenjang S1 di jurusan Matematika Institut Teknologi Sepuluh Nopember tahun 2012.

Di jurusan Matematika, penulis mengambil bidang minat Ilmu Komputer. Di ITS, penulis pernah menjadi staff HRD pada UKM Robotika, serta pernah menjadi anggota tim KRI (Kontes Robot Indonesia) untuk tim ITS, penulis juga pernah aktif berorganisasi menjadi Staff Kewirausahaan Paguyuban KSE ITS. Selama masa perkuliahan, penulis juga pernah mengikuti pertukaran pelajar selama 1 bulan di UTM (Universiti Teknologi Malaysia) dan 1 semester di *Dankook University*, Korea Selatan. Penulis memiliki *passion* yang cukup besar di bidang Ilmu Komputer terutama *formal verification, machine learning,* kriptografi, *data mining, artificial intelligence* dan *image processing*. Saat ini, penulis merupakan seorang CEO di CV. Indi Global yang merupakan perusahaan Konsultan IT di Surabaya.

Adapun mengenai informasi lebih lanjut atau ingin berdiskusi mengenai tugas akhir ini dapat ditujukan ke email penulis ikhwan12@mhs.matematika.its.ac.id atau ikhwan.ikhwan52@gmail.com.