Sains Indonesia Vol. 8, No.3 (2003) 27-33

Model Checking pada Logika Temporal Linear untuk Microwave

Yahma Wisnani

Departemen Matematika FMIPA, Universitas Indonesia

Abstrak

Makalah ini memperkenalkan model checking pada logika temporal linear serta aturan selama proses verifikasi berlangsung. Tahapan utama pada model checking adalah model, spesifikasi dan verifikasi. Tahapan model mengkonversikan sebuah rancangan menjadi sebuah model dalam bentuk struktur Kripke; tahapan spesifikasi merepresentasikan semua sifat yang harus dipenuhi oleh rancangan kebentuk bahasa logika temporal linear dan tahapan verifikasi membuktikan apakah spesifikasi telah terpenuhi sepanjang lintasan dalam model. Oven microwave digunakan sebagai contoh rancangan yang akan diverifikasi.

Abstract

Checking Model of Linear Temporal Logic for Microwave Oven: This paper introduce checking model of linear temporal logic and its role within the process. The main steps of model checking are modelling, specification and verification. The modelling step convert a design into a model in Kripke structure forms; the spesification step represent all properties of the satisfy design that it should be stated by using linear temporal logic, and the verification step determine that the specification should be hold along paths in the model. Microwave oven is used as a design exampel to be verify.

Keywords: formal method, linear tempral logic, Kripke structure, closure.

I. PENDAHULUAN

Sistem perangkat keras dan sistem perangkat lunak merupakan bagian terpenting pada rancangan sistem *Information and Communication Technologi* (ICT). ICT telah berkembang dengan pesatnya dan situasi ini menciptakan persaingan pasar yang ketat; faktor "*time*" dan "*money*"" merupakan variabel penentu untuk merebut pasar. Perusahaan sebagai produser berpacu dengan waktu dan menghadapi dilema antara memaksimalkan hasil produksi dan meningkatkan kwalitasnya tetapi meminimalkan biaya produksi; biaya produksi dapat ditekan jika *bug* dapat dideteksi sedini mungkin pada proses perancangan [1]. *Bug* atau kesalahan adalah kondisi yang disebabkan oleh ketidak sesuaian antara implementasi dan spesifikasi [2]. Metode formal merupakan jalan keluar pada proses verifikasi suatu rancangan.

Tidak terdeteksinya kesalahan pada perangkat lunak maupun perangkat keras setelah hasil produksi beredar dipasaran bukan saja berdampak buruk pada kelangsungan perusahaan atau pabrik yang memproduksinya bahkan juga menyangkut keamanan jiwa pemakainya.

Sebagai contoh adalah kesalahan yang pernah terjadi pada processor Pentium II buatan Intel pada bagian unit pembagi bilangan rasional menyebabkan Intel kehilangan dana sebesar US$ 475 milyar untuk menggantikan processor tsb; contoh lainnya adalah penundaan pembukaan Airport selama 9 bulan di Denver mengakibatkan perusahaan bersangkutan kehilangan dana sebesar US$1.1 milyar per hari yang diakibatkan oleh kesalahan perangkat lunak dalam sistem penanganan bagasi sehingga tidak dapat dilakukan sistem penanganan tiket secara *on-line*; hal ini menyebabkan perusahaan penerbangan yang bersangkutan terancam ditutup.

Masih kuat dalam ingatan kita adalah peristiwa meledaknya Rocket Arianne 5 setelah 36 detik diluncurkan pada tanggal 4 Juni 1996 yang mengakibatkan seluruh peserta misi dalam roket

kehilangan nyawanya; peristiwa ini terjadi karena perangkat lunak yang mengkonversi bilangan desimal dengan panjang 64 bit dikecilkan menjadi 16 bit. Kejadian fatal lainnya adalah tewasnya 6 pasien kanker antara tahun 1985 sampai tahun 1987 yang diakibatkan oleh kesalahan rancangan perangkat lunak pada bagian kontrol mesin terapi radiasi Therac-25 [3].

Beberapa teknik verifikasi untuk perangkat lunak adalah *peer review* dan testing, sedangkan untuk perangkat keras adalah emulasi, simulasi dan analisa structural [4]. Slogan *the sooner, the better* bagi semua tehnik diatas tidak terpenuhi karena tehnik tersebut membutuhkan waktu yang lama dan biaya besar karena pengecekan dilakukan oleh sejumlah pakar dibidangnya masing-masing.

Hasil penelitian menyebutkan bahwa waktu yang diperlukan untuk perancangan sistem komplek jauh lebih besar dari pada waktu untuk konstruksi [3]. Model checking adalah suatu teknik verifikasi terkini yang paling efektif karena bekerja hanya dengan memeriksa semua status yang mungkin dalam rancangan secara *brute force* [1]; sebagai contoh dimana model checking telah diterapkan adalah kasus *deadlock* dapat dideteksi pada sistem pemesanan tiket secara *online*; juga kesalahan pada modul-modul Deep Space I space-craft yang akhirnya sukses diluncurkan pada bulan Oktober 1998.

Tehnik verifikasi sistem dapat diklasifikasikan menurut kriteria [5]: *proof-based* dan *model-based* . Model checking adalah suatu verifikasi yang *fully-automatic model-based*, dikatakan demikian karena perancang langsung dapat mengetahui kebenaran atau menemukan kesalahan suatu rancangan tanpa perlu terlebih dahulu meng-eksekusi atau mempergunakan rancangan tersebut.

Model checking pada logika temporal linear dapat diterapkan jika sistem yang akan diteliti dapat direpresentasikan dalam suatu model M dengan status berhingga dalam bentuk *state transitions graph* (struktur Kripke) dan spesifikasi sifat-sifat formula lintasan φ yang direpresentasikan dalam logika temporal linear demikian sehingga dapat diverifikasi apakah suatu formula φ terpenuhi sepanjang lintasan π dalam model M? Notasi: $M,\pi \models \varphi$?

Translasi dari suatu deskripsi sistem menjadi sebuah struktur Kripke biasanya menimbulkan masalah yang disebut "*state-explosion problem* [6]".

Terdapat tiga (3) langkah utama dalam proses model checking [7], [4], [8], [9], [10] yaitu:

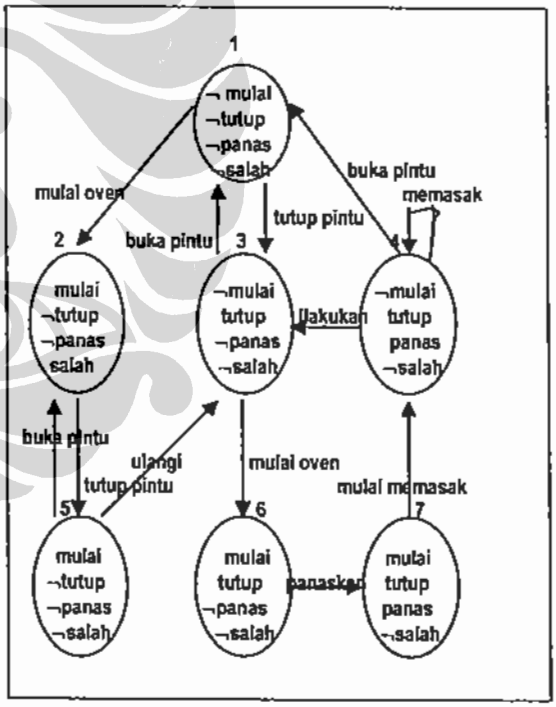
- Model
- spesifikasi
- verifikasi

2. MODEL

Langkah pertama pada proses model checking adalah mengkonversikan suatu rancangan sistem kedalam bentuk formal yang dapat diterima oleh alat-alat model checking. Struktur Kripke merupakan pilihan yang tepat untuk merepresentasikan suatu rancangan karena struktur kripke merupakan graf berarah.

Struktur Kripke didefinisikan sebagai berikut [8]: Suatu struktur Kripke $M = (Q, \delta, I, L)$ pada sebuah himpunan proposisi atomik A dimana Q adalah himpunan berhingga status yang tak kosong, $\delta = Q \rightarrow P(Q)$ adalah fungsi transisi, $I \subseteq Q$ adalah himpunan status awal dan $I: S \rightarrow P(A)$ adalah fungsi yang melabel setiap transisi dengan sebuah himpunan proposisi atomik dimana status bernilai benar (true) atau salah (false).

Sebuah status q adalah *predecessor* dari status q' dalam struktur Kripke M jika $q' \in \delta(q)$. Status q' disebut *successor* dari q. Sebuah transisi adalah setiap pasangan terurut (q, q') demikian sehingga $q' \in \delta(q)$.



***Gambar 1.** Contoh rancangan microwave [4].*

Struktur Kripke untuk oven microwave seperti yang terlihat pada Gambar 1 terdiri dari 7 status dengan 4 proposisi atomik yaitu {"mulai", "panas", "tutup" dan "pintu"} yang berarti:

- proposisi "mulai" berarti microwave mulai beroperasi
- proposisi "panas" berarti microwave dalam kondisi panas
- proposisi "tutup" berarti pintu microwave dalam kondisi tertutup
- proposisi "salah" berarti microwave dalam kondisi "*error*".

Ke-7 status tersebut dinamakan sebagai status 1, 2, 3, 4, 5, 6 dan 7 serta transisi antara sesama status juga terlihat pada Gambar 1. Status 1, 3, 4, 6 dan 7 serta transisi sesama status tersebut menunjukkan microwave beroperasi normal sebagaimana yang diharapkan oleh pemakai; sedangkan status 2 dan 3 beserta transisi diantara keduanya menunjukkan operasi yang tidak wajar sekaligus tidak diharapkan karena adanya kesalahan pada microwave yang ditunjukkan oleh proposisi "salah" dikedua status tersebut. Status 1 sebagai status awal menunjukkan microwave berada pada status "siap digunakan" atau siap diambil hasilnya oleh pemakai, karena pada status 1 berlaku proposisi tidak mulai, tidak tutup, tidak panas dan tidak salah; jika pintu microwave ditutup oleh pemakai yang ditunjukkan oleh transisi dari status 1 ke status 2 (direpresentasikan edge atau busur berarah dari status 1 ke status 2 maka microwave berada pada status 2 dimana berlaku proposisi tidak mulai, tutup, tidak panas dan tidak salah, proposisi tidak panas menunjukkan tidak ada proses pemanasan pada microwave; pemakai dapat kembali membuka pintu jika membatalkan atau memperbaiki tindakan sebelumnya seperti merubah lamanya waktu memasak atau menambah makanan yang akan dimasak pada microwave, jika tidak pemakai dapat menekan tombol mulai yang terdapat di tabung bagian depan microwave sehingga kini sistem berada pada status 6, secara otomatis sistem akan berpindah ke status 7 dilanjutkan ke status 4, selama perpindahan ini pemakai hanya pasif atau menunggu; pada status 4 terlihat busur berarah kedirinya sendiri sebagai transisi memasak, ini menunjukkan microwave sedang memasak, lamanya waktu memasak tergantung pada "tombol waktu" yang dipilih pemakai setelah pemakai menutup pintu microwave dan sebelum menekan tombol mulai; setelah waktu yang dipilih pemakai habis (biasanya ditandai dengan bunyi alarm) maka pemakai dapat segera membuka pintu dan mengambil hasilnya sehingga sekarang kembali ke status awal yaitu status 1; tetapi jika pemakai tidak segera membuka pintu (misalkan karena pemakai tertidur) maka sistem secara otomatis berpindah kembali ke status 3, pada situasi ini biasanya alarm berbunyi secara berkala sebagai pemberitahuan kepada pemakai bahwa proses memasak yang diinginkan pemakai telah selesai, jika pemakai membuka pintu maka microwave kembali ke status awal.

3. SPESIFIKASI

Tehnik verifikasi disebut sebagai *Temporal Logic model checking* dikembangkan pada tahun 1980-an secara terpisah oleh Clarke bersama Emerson di USA dan oleh Queille bersama Sifakis di Perancis [3]. Logika temporal sangat berguna untuk verifikasi formal. Semua sifat yang diharapkan harus dipenuhi oleh rancangan sistem diformulasikan dalam bentuk logika temporal. Logika temporal merupakan perluasan dari logika klasik dimana operatornya berelasi dengan faktor "waktu"; waktu yang lampau, waktu sekarang dan waktu yang akan datang. Logika klasik hanya mampu menguraikan kondisi statis karena hanya melibatkan operator Boolean ($\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$) dan operator first order ($\exists$,$\forall$) [9].

Sifat temporal dibagi menjadi 3 macam yaitu Linear Temporal Logic (LTL), Computational Tree Logic (CTL), dan CTL*. Dalam tulisan ini logika yang digunakan adalah LTL. LTL ditemukan oleh Prior (1960) dan untuk pertama kalinya digunakan oleh A. Pnueli dan Z. Manna [7] untuk menspesifikasi sifat-sifat pada sistem reaktif atau concurrent. Sifat LTL disebut juga sebagai *path formula* atau formula lintasan.

Sintak LTL

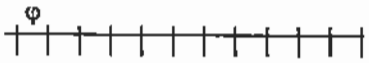
Definisi LTL [4],[1], [5], [9], [6]:

- Jika φ, φ_1, φ_2 formula LTL maka $X\varphi$, $F\varphi$, $G\varphi$, $\varphi_1 U \varphi_2$, $\varphi_1 W \varphi_2$, $\varphi_1 R \varphi_2$, juga formula LTL. X, F, G, U, W dan R disebut operator temporal.
- Jika φ_1, φ_2 formula LTL maka $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \rightarrow \varphi_2$, $\varphi_1 \leftrightarrow \varphi_2$, juga formula LTL.

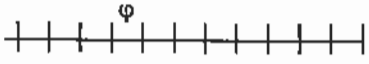
Ilustrasi untuk semantic pada LTL

Pendekatan logika temporal linear didefinisikan untuk menguraikan barisan linear dari status, jadi setiap status hanya mempunyai satu (1) *successor*.

- X ("next time") artinya sifat terpenuhi tepat oleh sebuah status berikutnya pada lintasan.

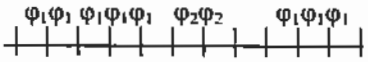


- F ("in the future atau eventualy") artinya sifat akhirnya terpenuhi disuatu status pada lintasan.



- G ("globally") artinya sifat terpenuhi disetiap titik pada lintasan.

- U ("until") memgkombinasikan dua (2) sifat, yang terpenuhi bila ada sebuah status pada lintasan dimana sifat kedua terpenuhi dan disetiap status sebelumnya pada lintasan sifat pertama terpenuhi. Misal: $\varphi = \varphi_1 U \varphi_2$



- W ("weak-until", "wait-for") adalah $G\varphi$ atau $\varphi = \varphi_1 U \varphi_2$
- R ("release") adalah logika kebalikan (*dual logic*) dari operator U, artinya sifat pertama terpenuhi bila sifat kedua terpenuhi sepanjang lintasan, mungkin juga sifat pertama tidak terpenuhi dimasa yang akan datang.

Semantik

Sebuah sistem memenuhi sebuah formula LTL φ jika dan hanya jika setiap lintasan dalam model sistem memenuhi φ, oleh karenanya formula LTL disebut juga sebagai formula lintasan. Semantik pada LTL diberikan kepada lintasan atau *trace* (berhingga atau tak berhingga status) [6]: $\pi=s_1s_2s_3$....., jika penulisan *suffix* i berarti lintasan dimulai dari status i ditulis π_i. contoh: $\pi_4=s_4s_5s_6$.....

Semantik pada LTL sbb: $\forall\varphi,\varphi_1,\varphi_2 \in AP$ [11], [8]:

- $\pi|=\varphi \leftrightarrow \varphi \in L(s_1)$, $\pi|=\neg\varphi \leftrightarrow \varphi \notin L(s_1)$
- $\pi|=\varphi_1 \wedge \varphi_2 \leftrightarrow \pi|=\varphi_1$ dan $\pi|=\varphi_2$
- $\pi|=\varphi_1 \vee \varphi_2 \leftrightarrow \pi|=\varphi_1$ dan $\pi|=\varphi_2$
- $\pi|=X\varphi \leftrightarrow \pi_2|=\varphi$
- $\pi|=F\varphi \leftrightarrow \exists i \geq 1 \ni \pi_i|=\varphi$
- $\pi|= G\varphi \leftrightarrow \forall i \geq 1 \ni \pi_i|=\varphi$
- $\pi|=\varphi_1 \cup \varphi_2 \leftrightarrow \exists i \geq 1 \ni \pi_i|=\varphi$, dan $\forall i\ 1 \leq j \leq i 1 \ni \pi_j|=\varphi_2$
- $\pi|=\varphi_1 W \varphi_2 \leftrightarrow \pi|= G\varphi$ atau $\pi|=\varphi_1 \cup \varphi_2$
- $\pi|=\varphi_1 R \varphi_2 \leftrightarrow \pi_k|=\varphi_2$ atau $\exists j,\ 1 \leq j \leq ik$ dan $\pi_j|=\varphi_1$.

Contoh sifat

Sebuah formula LTL bersifat global, artinya deskripsi suatu rancangan dimulai dengan kalimat: "setiap eksekusi pada suatu lintasan...." [4]. Sifat yang diharapkan dipenuhi oleh rancangan microwave pada suatu lintasan pada Gambar 1 antara lain:

1. "Sistem tidak akan pernah ke proses panas sampai pintu microwave dalam keadaan tertutup", formulanya dalam LTL sbb: φ = ¬panas ∪ tutup. Artinya untuk setiap eksekusi terdapat suatu lintasan dimana proses panas tidak akan dicapai sebelum pintu microwave tertutup".
2. "sistem selalu berada pada kondisi error atau tidak error"; formulanya dalam LTL adalah: φ=error∨¬error.

4. VERIFIKASI

Akan dibahas lebih dulu tentang klosur, atom serta graf untuk atom pada semua status untuk suatu formula LTL.

Klosur

Definisi: *Closure* (klosur) dari formula LTL φ (ditulis CL(φ)) adalah himpunan terkecil dari formula LTL yang memenuhi [4], [1], [9]:

1. $\varphi \in CL(\varphi)$
2. Jika $\varphi \in CL(\varphi)$ maka $\neg\varphi \in CL(\varphi)$
3. Jika $\varphi_1 \vee \varphi_2 \in CL(\varphi)$ maka $\varphi_1, \varphi_2 \in CL(\varphi)$
4. Jika $X\varphi \in CL(\varphi)$ maka $\varphi \in CL(\varphi)$
5. Jika $\neg X\varphi \in CL(\varphi)$ maka $X\neg\varphi \in CL(\varphi)$
6. Jika $\varphi_1 \cup \varphi_2 \in CL(\varphi)$ maka $\varphi_1, \varphi_2, (\varphi_1 \cup \varphi_2) \in CL(\varphi)$.

Catatan:

Menurut definisi no.2 jika $\varphi \in CL(\varphi)$ maka $\neg\varphi$, $\neg\neg\varphi$, $\neg\neg\neg\varphi$, ...$\in CL(\varphi)$ tetapi untuk pembahasan dalam tulisan ini hanya diwakili oleh $\neg\varphi \in CL(\varphi)$.

Gambar 2 adalah algoritma untuk mencari klosur suatu formula LTL φ.

```
1.  CL = {φ} ;
    K = {φ} ;
    while K ≠ Ø do
2.      pilih φ ∈ K dan keluarkan dari K;
        if φ≠¬φ' dan φ'∉CL(φ) maka
3.          tambahkan φ' pada CL dan K;
        if φ =φ1∨φ2 maka
4.          if φ1∉CL maka tambahkan φ1
                pada CL dan K;
5.          if φ2∉CL maka tambahkan φ2
                pada CL dan K;
6.      if φ=Xφ' dan φ'∉CL maka tambahkan
            φ' pada CL dan K;
        if φ =¬X¬φ' dan φ'∉CL maka
7.          tambahkan φ' pada CL dan K;
        if φ =φ ∪φ2 maka
8.          if φ1∉CL maka tambahkan φ1
                pada CL dan K;
            if φ2∉CL maka tambahkan φ2
                pada CL dan K;
    end while;
    return CL;
```

Gambar 2. Algoritma untuk mencari klosur suatu formula LTL φ.

Mengacu pada struktur Kripke untuk microwave pada gambar 1. Misalkan akan dicari klosur dari formula CL(ψ) = CL(¬φ) = ¬((¬panas ∪ tutup). Dengan mengaplikasikan algoritma pada gambar 2 maka dihasilkan klosur dengan langkah seperti yang terlihat pada gambar 3 sbb: CL(ψ)=CL(¬φ) = {¬φ, φ, Xφ, ¬Xφ, X(¬φ), ¬X¬φ, panas, tutup, ¬panas ¬tutup}

No:	CL(¬φ), menurut gambar 2 pada:
1	baris 1 didapat φ
2	baris 2 didapat ¬φ
3	baris 5 didapat Xφ
4	baris 2 didapat ¬Xφ,
5	baris 6 didapat X(¬φ)
6	baris 2 didapat ¬X¬φ
7	baris 7didapat "panas"
8	baris 8 didapat "tutup"
9	baris 2 didapat "¬panas"
10	baris 2 didapat "¬tutup"

Gambar 3. Langkah-langkah klosur.

Atom

Sebuah atom pada struktur Kripke M = (Q, δ, I, L) dan sebuah himpunan proposisi atomik AP dan sebuah formula LTL φ adalah pasangan terurut $A=(q_A,K_A)$ dengan $q_A \in Q$ dan $q_A \subseteq CL(\varphi) \cup AP$ demikian sehingga seperti yang terlihat pada gambar 4.

1. $\forall\ p \in AP$ maka $p \in K_A \leftrightarrow p \in I(q_A)$
2. $\forall\ \psi \in CL(\varphi), \psi \in K_A \leftrightarrow \neg\psi \notin K_A$
3. $\forall\ \varphi_1 \vee \varphi_2 \in CL(\varphi),\ \varphi_1 \vee \varphi_2 \in K_A \leftrightarrow \varphi_1 \in K_A$ dan $\varphi_2 \in K_A$
4. $\forall\ \neg X\varphi \in CL(\varphi), \neg X\varphi \in K_A \leftrightarrow X\neg\varphi \in K_A$
5. $\forall\ \varphi_1 \cup \varphi_2 \in CL(\varphi),\ \varphi_1 \cup \varphi_2 \in K_A \leftrightarrow \varphi_1$ atau $\varphi_2 \in K_A$, $X(\varphi_1 \cup \varphi_2) \in K_A$.

Gambar 4. Model atom pada struktur Kripke.

Atom (s_A,K_A) mendefinisikan suatu himpunan maksimal yang anggotanya sesuai dengan s_A juga sesuai satu dengan lainnya; sebuah status dimungkinkan berasosiasi dengan banyak K_A [5].

Contoh atom

Akan dicari K_A untuk setiap status A dari formula φ=¬panas∪tutup pada rancangan model M seperti yang terlihat pada Gambar 1; anggota himpunan atom (s_A,K_A) hanya bergantung pada proposisi atomik "tutup" dan "panas", bukan pada "mulai" dan "salah".

- misal $\varphi=\varphi_1 \cup \varphi_2 = \neg panas \cup tutup \in K_A$;

status 1 pada Gambar 1 mempunyai proposisi ¬tutup dan ¬panas maka menurut baris 1 pada gambar 4 berlaku:

- $\neg tutup \in K_A$ dan
- $\neg panas \in K_A$
- misalkan $\varphi_1=\neg tutup$ dan $\varphi_2=\neg panas$ sehingga berdasarkan definisi atom baris 5 pada gambar 4 berlaku $\varphi_2=\neg panas \in K_A$ dan $X\varphi \in K_A$ karena $\varphi \in CL(\varphi)$ (baris 1 pada gambar 3) dan $\varphi \in K_A$ (dimisalkan)
- $\neg X\neg\varphi \in K_A$ (menurut baris 2 pada Gambar 4) karena $\neg X\neg\varphi \in CL(\varphi)$ (baris 6 pada gambar 3) dan $X\neg\varphi \notin K_A$

Sehingga untuk $\varphi \in K_A$ berlaku:

$K'_1 = \{\neg tutup, \neg panas, \varphi, X\varphi, \neg X\neg\varphi\}$

Dengan cara yang sama untuk $\varphi \notin K_A$ dihasilkan:

$K''_1 = \{\neg tutup, \neg panas, \neg\varphi, X\neg\varphi, \neg X\varphi\}$

Status 2 juga mempunyai proposisi yang sama dengan status 1 sehingga dapat dituliskan:

$K'_{1,2} = \{\neg tutup, \neg panas, \varphi, X\varphi, \neg X\neg\varphi\}$

$K''_{1,2} = \{\neg tutup, \neg panas, \neg\varphi, X\neg\varphi, \neg X\varphi\}$

Status 3, 5 dan 6 mengandung proposisi tutup dan tidak panas sehingga dengan cara yang sama didapat:

$K'_{3,5,6} = \{tutup, \neg panas, \varphi, X\varphi, \neg X\neg\varphi\}$

$K''_{3,5,6} = \{tutup, \neg panas, \neg\varphi, X\neg\varphi, \neg X\varphi\}$

Status 4 dan 7 sama-sama memiliki proposisi tutup dan panas sehingga:

$K'_{4,7} = \{tutup, panas, \varphi, X\varphi, \neg X\neg\varphi\}$

$K''_{4,7} = \{tutup, panas, \neg\varphi, X\neg\varphi, \neg X\varphi\}$.

Graf untuk atom

Dari himpunan atom (s_A,K_A) dapat dibuat suatu graf G [4], [6] dimana node mewakili setiap atom dan setiap node dalam struktur graf G terhubung satu dengan lainnya melalui operator "X" menurut aturan:

- jika suatu status "i" terhubung dengan status "j" ($i \neq j$ atau i=j) dalam model M maka atom-atom dari status tersebut juga saling terhubung.
- suatu status dalam model M dapat mempunyai lebih dari satu atom sesuai dengan banyaknya keterhubungan status tersebut dalam model M.

Terdapat terdapat 14 node sebagai atom dengan 22 busur pada graf beraraf G, tetapi untuk mudahnya cukup direpresentasikan sebagai berikut:

Pada Gambar 1 status 1 terhubung dengan status 2 dan status 3 maka atom $(1,K'_1)$ baris 1 dan baris 2 pada Gambar 5 memperlihatkan *edge* atau busur dari node $(1,K''_1)$ ke node (i,K'_i) i=1,2 sebagai konsekwensi dari formula Xφ dalam atom $(1,K'_1)$ menunjukkan node $(1,K'_1)$ terhubung dengan node $(2,K'_2)$, $(3,K'_3)$ dan $(3,K''_3)$ karena atom $(2,K'_2)$, $(3,K'_3)$ dan $(3,K''_3)$ beranggotakan formula φ. Sedangkan pada baris 2 node $(1,K''_1)$ yang mengandung formula X¬φ hanya

terhubung ke node $(2,K''_2)$ karena $(2,K''_2)$ $(2,K''_2)$ beranggota $\neg\varphi$ dan atom $(2,K''_2)$ tidak lagi terhubung dengan atom lainnya.

no	atom
1	$(1,K'_1) \rightarrow (2,K'_2), (3,K'_3), (3,K''_3)$
2	$(1,K''_1) \rightarrow (2,K''_2)$
3	$(2,K'_2) \rightarrow (5,K'_5), (5,K''_5)$
4	$(3,K'_3) \rightarrow (1,K'_1), (6,K'_6), (6,K''_6)$
5	$(3,K''_3) \rightarrow (1,K''_1)$
6	$(4,K'_4) \rightarrow (1,K'_1), (3,K'_3), (3,K''_3)$
7	$(4,K''_4) \rightarrow (1,K''_1)$
8	$(5,K'_5) \rightarrow (2,K'_2), (3,K'_3), (3,K''_3)$
9	$(5,K''_5) \rightarrow (2,K''_2)$
10	$(6,K'_6) \rightarrow (7,K'_7), (7,K''_7)$
11	$(7,K'_7) \rightarrow (4,K'_4), (4,K''_4)$

Gambar 5. Edge atau busur dari node ke node.

Proses ini dilanjutkan untuk status lainnya dengan memperhatikan hubungan setiap status A dengan status lainnya pada Gambar 1 dan setiap atom (s_A,K_A) dimana atom yang beranggotakan formula "X(φ)"adalah sebagai *predecessor* dan atom yang beranggotakan formula φ menjadi *successor*.

Emptiness

Algoritma model checking pada Gambar 6 mengembalikan *true* jika dan hanya formula LTL φ dipenuhi di semua eksekusi pada model M, atau untuk mudahnya algoritma diperiksa dengan negasinya yaitu: jika terdapat suatu eksekusi pada M dengan formula $\neg\varphi$, apakah implementasi algoritma tidak dipenuhi atau *false*?.

Algoritma model checking pada LTL

Input: sebuak struktur Kripke M dan formula LTL φ;
Output: *true* jika terdapat suatu exekusi pada M (dimulai dari status awal) yang memenuhi φ; else *false*

Algoritma:
1. Cari klosur CL(φ);
2. Cari sebuah himpunan atom A pada M dan φ;
3. Gambarkan graf G dimana node sebagai atom dari A, bila $A=(S_A,K_A)$ dan $B=(S_B,K_B)$ dan (A,B) adalah busur pada $G \leftrightarrow s_B \in \delta(s_A)$ dan ∀ formula $X\varphi \in CL(\varphi)$ maka $X\varphi \in K_A \leftrightarrow \varphi \in K_A$;
4. if ∃ lintasan dimana Fφ terpenuhi dalam G return true;
   else return false;

Gambar 6. Algoritma model checking.

Contoh:
Model M adalah rancangan microwave pada Gambar 1 dengan sifat "microwave tidak pernah panas sampai pintu microwave tertutup" diformulasikan dalam formula LTL $\varphi=\neg$panas$\cup$tutup. Akan diperiksa dengan negasi yaitu: terdapat suatu eksekusi pada M dengan $\psi=(\neg\varphi)=\neg((\neg$panas$)\cup$tutup)) apakah algoritma model checking pada Gambar 6 mengembalikan *false*?, atau apakah terdapat suatu lintasan dimana $\neg\varphi$ tidak terpenuhi?

Aplikasikan algoritma pada Gambar 6 dengan tahapan sbb:

1. klosur $CL(\psi)=CL(\neg\varphi)$ = {$\neg\varphi$, φ, Xφ, $\neg X\varphi$, $X(\neg\varphi)$, $\neg X\neg\varphi$, panas, tutup, ¬panas ¬tutup} seperti yang terlihat pada Gambar 3.
2. Atom A untuk setiap status pada Gambar 1 adalah: (i,K'_i) dan (i,K'_i), i=1,2,...,7. seperti yang diuraikan pada bagian contoh atom.
3. Graf G yang merupakan representasi dari keterhubungan seluruh atom dari formula $\neg\varphi$ diwakili seperti yang direpresentasikan dalam pada Gambar 5.
4. Formula $\psi=\neg\varphi$ terpenuhi jika dan hanya jika terdapat lintasan π yang dimulai dari $(1,K''_1)$ ke atom lainnya dimana formula formula $\psi=\neg\varphi$ terpenuhi dalam G. Pada Gambar 5 lintasan π yang dimulai dari $(1,K''_1)$ hanya terhubung ke node $(2,K''_2)$ dan tidak berlanjut ke atom lainnya dari $(2,K''_2)$, atom ini menunjukkan tidak terdapat suatu lintasan π yang memenuhi formula ψ dalam G, Jadi tidak ada eksekusi pada M yang memenuhi ψ. Akibatnya setiap eksekusi pada M memenuhi formula φ. Terbukti bahwa tidak mungkin microwave menjadi panas sebelum pintu microwave dalam kondisi tertutup.

Biasanya algoritma menyediakan *counterexample* bila formula tidak dipenuhi oleh rancangan sistem. *Counterexample* terdiri dari sebuah eksekusi pada sistem yang tidak dipenuhi sehingga pemakai dapat memperbaiki rancangan sistem mula-mula [1].

Kompleksitas algoritma model checking pada LTL adalah PSPACE-complete [11].

5. KESIMPULAN

- Model checking mudah digunakan karena sifat rancangan dapat di chek secara individu.
- Model checking adalah pendekatan verifikasi yang umum karena berdasarkan pada teori elementer pada algoritma graf, struktur data dan logika.

- Model checking dapat menghemat biaya dan waktu karena proses model checking dilaksanakan pada tahap perancangan sistem.

Penelitian lanjutan

- Model checking pada LTL menggunakan automaton Büchi.
- Model checking pada CTL dan CTL*.

DAFTAR PUSTAKA

[1]. Katoen, Joost-Pieter. [2002]. *System Validation--Linear Temporal Logic*. Formal Method and Tools Group.

[2]. Juhan Ernits. [2004] *Introduction to formal verification: Verification of software*. Lecture notes. Aarthus University. Denmark.

[3]. Jörg Niere., Chandra Kurnia Jaya. [2004]. *Verification, Validation und Testen von Sicherheitskrischen Sisteme*. Universität Siegen, Fachgruppe für Praktische Informatik.

[4]. James Hook dan Sangtao Xia. [2004]. *Model checking for Bug Discovery*. CSE 506/606, OGI School.

[5]. Lange, Martin. [2002]. *Games for Modal and Temporal Logic*. Doctor of Philisophy. Unuversity of Edinburg. Edinburg.

[6]. Tom Henzinger. [2004]. *Four Lectures on Model Checking*. University of California, Berkeley.

[7]. Dragana Cvijanovic. [2004]. *Model Checking*. 3C05.

[8]. Matt Dwyer, John Hatcliff, Robby. [2002]. *Specifiction and Verification of Reactive Sistems*. CIS 842. Kansas State University . Kansas.

[9]. Otmar Onderek. [2004]. *Introduction to Temporal Logics and Model Checking*. Department of Computer Science, Technical University Ostrava, Chech Republik.

[10]. Yuguo Sun. [2003]. *Model checking, A Tutorial Introduction*. Seminar: Sicherheitskritische Systeme.

[11]. Heijo Heljanko. *Model Checking with Finite Complete Prefixes is PSPACE complete*. [2000]. Helsinki University of Technology Laboratory for Theorical Computer Science. Helsinki.