

External Embedding: Sebuah Metode untuk Verifikasi Program¹

Ade Azurat
Fakultas Ilmu Komputer
Universitas Indonesia

E-mail: ade@cs.ui.ac.id

Abstrak

Kebutuhan Teknologi Computer Aided Verification dalam industri perangkat lunak jelas tidak diragukan lagi, namun metode-metode yang terdapat didalamnya masih memiliki banyak keterbatasan sehingga penerapannya secara umum masih belum terwujud. Artikel ini mempresentasikan sebuah metode verifikasi perangkat lunak yang disebut external embedding menggunakan Theorem Prover HOL. External embedding menghindari kompleksitas dari metode deep embedding dan ketidaklengkapan dari metode shallow embedding. Operasi pembuktian bekerja pada representasi abstract syntax tree dari sebuah program dan spesifikasinya. Metode ini telah diterapkan pada alat bantu verifikasi program yang disebut xMech dan juga untuk alat bantu verifikasi aplikasi database yang disebut LinguHOL.

Kategori: *program verifikasi, rekayasa perangkat lunak, metode formal.*

Permasalahan

Computer Aided Verification (CAV) adalah sebuah teknologi yang menggabungkan antara kehandalan pendekatan formal dan algoritma pembuktian otomatis. Manfaat dari teknologi tersebut sudah cukup dipahami [12]. Namun sayangnya belum ada sebuah alat bantu dengan teknologi tersebut yang dapat digunakan dan cocok secara umum untuk kebutuhan industri perangkat lunak saat ini. Hal ini lebih disebabkan karena tingkat kesulitan dan kurangnya pemahaman mengenai spesifikasi formal.

CAV membutuhkan spesifikasi program yang formal. Padahal, sebuah kalimat spesifikasi yang informal bila diterjemahkan dalam spesifikasi formal dapat terdiri dari beberapa baris formula. Hal ini terjadi karena pada spesifikasi formal segala sesuatunya harus dinyatakan dengan sangat detil dan lengkap sehingga sulit untuk dikerjakan. Sebagai contoh lihat penelitian dari Prasetya [6] dan Vos [7] serta de Pol [4].

Pendekatan dalam verifikasi berbantuan komputer (CAV)

Pada studi literatur [3] dapat ditemukan tiga kategori teknik pendekatan untuk *CAV*, yaitu :

1. Verifikasi berdasarkan semantik dari bahasa. Teknik ini bekerja sebagai berikut: Menerjemahkan program kedalam formalisasi yang memiliki semantik dari bahasa pemrograman tersebut. Kemudian kebenaran program dinyatakan langsung dalam formalisasi tersebut dan pembuktiannya

dilakukan menggunakan aturan dari semantik bahasa. Keuntungan dari pendekatan ini adalah: formalisasi yang telah dilengkapi dengan alat bantu pembuktian dapat digunakan tanpa tambahan. Namun sebagai konsekuensinya diperlukan proses transformasi yang juga harus diverifikasi. Kelemahan utama dari teknik ini adalah: Kalimat spesifikasi dari proses verifikasi menjadi cukup rumit. Kerumitan tersebut disebabkan oleh abstraksi yang harus dilakukan di tingkat pembuktian semantik untuk tiap langkah verifikasi. Padahal semantik dari sebuah bahasa relatif sudah rumit.

2. *Verification Condition Generation (VCG)*. Teknik ini merupakan teknik klasik dalam program verifikasi. Berdasarkan pada sebuah formalisasi logika, sebuah spesifikasi program direduksi menjadi kondisi verifikasi dalam bentuk formula logika yang harus dibuktikan. Keuntungan teknik ini adalah: Aspek yang bergantung pada bahasa pemrograman dapat dieliminasi secara otomatis sehingga mengurangi beban verifikasi. Selain itu, proses verifikasi dapat dilakukan menggunakan alat bantu yang standard. Kelemahannya adalah: Pada kenyataannya, kondisi verifikasi yang dihasilkan sangat besar dan rumit. Bila verifikasi kondisi tersebut tidak dapat dibuktikan, maka sulit untuk dapat mengetahui dari bagian program mana verifikasi kondisi tersebut berasal, sehingga sulit mengetahui bagian program yang perlu diperbaiki. Kelemahan yang lain adalah, proses verifikasi dilakukan setelah program selesai dibangun sehingga tidak bisa sejalan dengan pengembangan program.

¹ Artikel serupa yang lebih detil telah diterbitkan berbahasa inggris dengan judul *A framework for verification of concurrent systems* dalam *Proceedings of The 5th Indonesian Student's Scientific Meeting ISSM 2000, Paris, France*, Penulis: Ade Azurat. ISSN : 0855-8692, Halaman 293-296, Oktober 2000[5].

3. Verifikasi program yang interaktif. Teknik ini mengaplikasikan ide dari *Theorem Prover* pada logika pemrograman. *Theorem prover* diharapkan dapat menganalisa program seperti melakukan eksekusi dari program tersebut sambil melakukan verifikasi dari spesifikasi secara interaktif. Keuntungan dari teknik ini adalah: Pembuktian dapat dibuat dalam program sebagai anotasi (catatan) dalam program. Tahapan *Strengthening* dan *weakening* dari *pre-post* kondisi dapat diterapkan secara interaktif sesuai kebutuhan. Intuisi dalam pemrograman dapat langsung digunakan untuk melakukan pembuktian dan dapat lebih mudah mendeteksi kesalahan program dari pembuktian yang tidak berhasil. Kelemahannya adalah: *Theorem Prover* yang digunakan harus 'mengerti' akan bahasa pemrograman yang digunakan. Hal ini biasa disebut sebagai proses *embedding*. Konsekuensinya adalah alat bantu yang digunakan akan bergantung dan khusus untuk bahasa pemrograman tertentu saja.

Alat bantu verifikasi *xMech*[1] dan *LinguHOL*[9,11] yang dikembangkan menggunakan hasil dari penelitian ini dapat dikategorikan menggunakan teknik nomor tiga yaitu Verifikasi program yang interaktif. Namun sedikit berbeda dengan teknik dasar, proses *embedding* yang dilakukan menggunakan metode baru yang lebih flexible. Metode baru tersebut selanjutnya dinamakan *external embedding*[8,10]. Alasan utama penerapan metode ini adalah: Pembuktian dapat dilakukan langsung secara independen dengan bahasa pemrograman dari *Theorem Prover* yang digunakan.

External Embedding

Untuk membuktikan sebuah program dengan *theorem prover* seperti HOL[2], maka HOL harus 'mengerti' bahasa pemrograman yang digunakan. Dengan kata lain bahasa pemrograman tersebut harus di-*embed* terlebih dahulu di HOL. HOL adalah sebuah *theorem prover* yang biasa juga disebut sebagai *mechanized proof assistant* untuk *Higher order logic*. *Higher order logic* adalah logika predikat dengan penambahan tiga hal, yaitu: variable dapat menyatakan fungsi dan predikat, logika yang digunakan memiliki *type*, dan tidak ada pemisahan sintak antara *term* dan *formula*[2]. Dengan *embedding*, HOL akan mengenali program dan mengenali langkah pembuktian yang benar untuk program yang dimaksud.

Embedding dapat dijelaskan sebagai berikut: Sebuah *embedding* bahasa pemrograman L (bersama logika spesifikasi S) dalam *Theorem Prover* T ; artinya adalah menambahkan T , sehingga menggunakan aturan-aturan di T dapat melakukan analisa pembuktian S pada program berbahasa L .

Ada dua metode *embedding* yang umum dilakukan yaitu: *shallow embedding* dan *deep embedding*. Metode pertama mendefinisikan konstanta baru pada T yang merepresentasikan bahasa L . Definisinya dinyatakan

sebagai fungsi yang menyatakan semantik dari bahasa L . Metode kedua mendefinisikan tipe-data (*data-type*) dalam T . Semantik bahasa L pada T dinyatakan sebagai fungsi rekursif dari tipe-data yang didefinisikan tersebut.

Shallow embedding membuat pembacaan program dan spesifikasi kedalam logika menjadi lebih mudah. Namun sulit menyatakan aspek sintak di dalam logika sehingga terbatas untuk melakukan optimisasi khusus dalam program yang melibatkan penyesuaian sintak. *Deep embedding* memberikan kebebasan lebih. semakin dalam pendefinisian bahasa yang dilakukan maka semakin luas kebebasan yang dimiliki. Namun sejalan dengan meluasnya kebebasan tersebut maka kerumitan dari definisi dan proses verifikasi yang dihasilkan menjadi meningkat pula.

External embedding mengambil keuntungan dari bahasa pemrograman ML yang digunakan untuk mengimplementasikan *Theorem Prover* HOL. Untuk bahasa pemrograman L dan logika spesifikasi S , keduanya akan di-*embed* secara *shallow* pada HOL, namun sintak dari L dinyatakan dalam ML bersama dengan *parser* dan *pretty printers* dari L . *Parser* dan *Pretty printers* ini diperlukan agar pengguna tak perlu melihat internal formula yang biasanya sulit untuk dibaca. Saat proses pembuktian dimulai, bagian *embedding* di ML melakukan transformasi dari pembuktian tersebut dalam representasi *shallow embedding* kemudian memanggil taktik pembuktian dalam HOL dan mengembalikan hasilnya ke level *embedding* di ML . Karena sintak dari L , ada dalam ML , maka informasi tersebut dapat digunakan untuk menganalisa program yang ditulis dalam bahasa L tersebut secara langsung di ML . Hal ini dapat dikategorikan sebagai *unsafe*, karena proses analisa terjadi diluar HOL yang dijamin *soundness*-nya. Konsekuensinya adalah proses analisa langsung ini perlu dilakukan hati-hati. Walaupun demikian, umumnya proses analisa sintak ini relatif sederhana, sehingga akurasi bisa lebih mudah dipertanggungjawabkan.

Proses Verifikasi

Proses verifikasi menggunakan *external embedding* memiliki tiga bagian, yaitu:

- Logika. Bagian ini berisikan logika dasar pada *Theorem Prover* HOL dan logika spesifikasi bahasa dari program yang akan diverifikasi.
- Transformasi. Bagian ini menyediakan penyambung (antar muka) antara *external embedding* di HOL dengan alat bantu verifikasi lainnya termasuk dengan ML . Bagian ini terdiri dari beberapa fungsi yang melakukan transformasi yang diinginkan.
- *Proof Management*. Bagian ini menyediakan alat bantu untuk mengatur pembuktian yang dilakukan. Termasuk didalamnya adalah pembacaan berkas, pengaturan proses pembuktian, pemanggilan teorema-teorema yang dibutuhkan dan hal-hal lain.

Tahapan dalam proses verifikasi menggunakan HOL dapat dirangkum sebagai berikut:

1. Pembacaan input program dan spesifikasinya dengan *parser*.
2. Transformasi program dan spesifikasi dalam *Theorem Prover*.
3. Pembuktian dalam HOL menggunakan standard taktik yang dimiliki.
4. Melakukan reduksi melalui *deep embedding* di ML.
5. *Pretty print* hasil reduksi di ML ke HOL.
6. Penyelesaian pembuktian dalam HOL.

Penutup

Metode *external embedding* ini telah berhasil diterapkan dalam pengembangan alat bantu verifikasi *xMech* [1] dan *LinguHOL*[9,11]. *xMech* merupakan alat bantu verifikasi untuk program *imperative* dan *concurrent*. *LinguHOL* merupakan alat bantu verifikasi untuk aplikasi database.

Metode *external embedding* didasari pada transformasi antara *deep embedding* di ML dengan *shallow embedding* di HOL. *Theorem prover* HOL merupakan sistem logika yang didasari oleh sejumlah kecil aksioma dasar dan aturan inferensi sehingga *soundness* dapat dijamin[2]. Disisi lain, ML adalah sebuah bahasa pemrograman besar sehingga transformasi dan reduksi yang dilakukan di ML tidak bisa dijamin. Agar dapat meningkatkan jaminan kebenaran maka setiap tranformasi dan reduksi yang dilakukan di ML harus diuji konsistensinya terhadap sebuah “meta model”. Hal ini masih merupakan bagian penelitian yang perlu diselesaikan.

Referensi

- [1]. A. Azurat and I.S.W.B. Prasetya. *A preliminary report on xmech*. Technical Report UUCS-2002-008, Institute of Information and Computing Sciences Utrecht University, P.O.Box 80.089 3508 TB Utrecht The Netherlands, January 2002.
- [2]. M.J.C. Gordon and T.F. Melham, *Introduction to HOL*. Cambridge University Press, 1993.
- [3]. J.Meyer and A. Poetzsch-Heffter, *An Architecture for Interactive Program Provers*, p 63-77 in S.Graf and M.Schwartzbach, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2000)*, volume 1785 of Lecture Notes of Computer Science, Berlin Germany, 2000.
- [4]. J,van de Pol, J. Hooman and E. de Jong, *Formal Requirements Specification for Command and Control Systems*, Report 99-09, Computer Science Reports, Eindhoven University of Technology, the Netherlands, 1999.
- [5]. Ade Azurat. A framework for verification of concurrent systems. In *Proceedings of The 5th Indonesian Student's Scientific Meeting ISSM 2000, Paris - France*, ISSN : 0855-8692, pages 293-296, October 2000.
- [6]. I.S.W.B. Prasetya, *Mechanically Supported Design of Self-stabilizing Algorithms*. PhD thesis, Utrecht University, 1995.
- [7]. Tanja Vos, *Unity in Diversity: A Stratified Approach to the Verification of Distributed Algorithm*. PhD thesis, Utrecht University, 2000.
- [8]. A. Azurat and I.S.W.B. Prasetya. *A survey on embedding programming logic in a theorem prover*. Technical Report UU-CS-2002-007, Institute of Information and Computing Sciences Utrecht University, P.O.Box 80.089 3508 TB Utrecht The Netherlands, January 2002.
- [9]. A. Azurat, I.S.W.B. Prasetya, T.E.J. Vos, H. Suhartanto, B. Widjaja, L.Y. Stefanus, R. Wenang, S. Aminah, and J. Bong, *Towards Automated Verification of Database Scripts*,(dalam proses review paper *International Conference TPHOLS 2005 dalam emerging trend track*), 2005.
- [10]. A. Azurat, I.S.W.B. Prasetya, and S.D. Swierstra, *Embedding Programming Logics in HOL Theorem Prover*, *Jurnal Ilmu Komputer dan Teknologi Informasi*. Volume 2, Nomer 1, Mei 2002, ICIS, Indonesia.
- [11].H. Suhartanto, B. Widjaja, L.Y. Stefanus, S. Aminah, J. Bong, ISWB Prasetya, A. Azurat, *Developing Technology for Specifying and Generating Critical Data Processing Programs: End of Year RUTI Report 2003*, Fasilkom UI, 2004.
- [12].Clarke E.M., Wing J.M, *Formal Methods: State of the Art and Future Directions*. ACM Computing Surveys, 28(4):626-643, 1996.