Liet. matem. rink, **46***, spec. nr.*, 2006, 103–106

Induktyvinis metodas komponentinių programų sintezėje

Vaidas GIEDRIMAS (MII)
el. paštas: vaigie@fm.su.lt

Reziumė. Automatinė programų sintezė – aktualus programų sistemų inžinerijos mokslinių tyrimų objektas. Straipsnyje aprašomos induktyviojo metodo panaudojimo automatinėje komponentinių programų sintezėje galimybės. Induktyvusis metodas pristatomas kaip efektyvi pagalbinė priemonė sprendžianti pagrindines komponentinių programų sintezės sistemos, naudojančios struktūrinės sintezės metodą [5, 6, 7], problemas.

Raktiniai žodžiai: komponentinės sistemos, programų sintezė, indukcija.

Įvadas

Formalieji metodai programų sistemų inžinerijoje naudojami programų specifikavimui, kūrimui ir verifikavimui. Juos naudojančių sintezės sistemų privalumas – galimybė kurti „teisingas“, specifikaciją atitinkančias programas be klaidų. Be to, tokios sistemos leidžia kurti programas „iš viršaus žemyn“, programas specifikuojant deklaratyviuoju o ne imperatyviuoju būdu [11]. Sintezės sistemose (pvz.: Amphion, KIDS, NUT) dažniausiai naudojami deduktyvinis ir transformacinis metodai.

Kaip pabrėžiama [4, 5, 6], dauguma sintezės sistemų yra skirtos struktūrinėms ir objektinėms, tačiau ne komponentinėms programoms generuoti. Darbuose [6, 7] aprašoma deduktyvinį (struktūrinės sintezės [16]) metodą naudojanti komponentinių programų sintezės sistema. Šio straipsnio tikslas – nustatyti induktyvinio metodo (konkrečiai – induktyviojo loginio programavimo) panaudojimo galimybes komponentinių programų sintezės procese.

Komponentinių programų sintezės problemos

Nustatyta, kad naudojant komponentinių programų sintezės sistemą, naudojančią struktūrinės sintezės metodą, susiduriama su šiomis problemomis [5]:

- **Specifikacijos problema**. Jei vartotojo pateikta specifikacija yra netiksli arba su klaidomis, sintezės rezultatas – nekorektiškas.
- **Neapibrėžtųjų komponentų problema**. Komponentinių programų sistemų gyvavimo ciklas skiriasi nuo struktūrinių programų gyvavimo ciklo. Komponentinės programų sistemos dažniausiai kuriamos pagal principą *Išrink-Pritaikyk-Bandyk* (*angl. Select-Adapt-Test*), todėl kūrimo pradžioje dažniausiai nėra žinomi visi sistemai sukurti reikalingi komponentai. Tuo tarpu naudojant struktūrinės sintezės metodą, daroma prielaida, kad visos aksiomos turi jas realizuojančius

komponentus ir reikia tik išspręsti jų pasirinkimo uždavinį. Sintezės sistemai neradus reikiamos aksiomos (o kartu ir komponento) pranešama, kad uždavinys neišsprendžiamas. Sintezės sistema negali nustatyti, kokių konkrečiai aksiomų ir kokių konkrečiai komponentų trūksta.

- **Nefunkcinių reikalavimų problema**. Darbe [6] aprašyta komponentinių programų sintezės sistema suteikia galimybę naudotojui specifikuoti tik funkcinius reikalavimus.

Siekiant išspręsti šias problemas tikslinga nagrinėti kitų generavimo metodų panaudojimo komponentinėms sistemoms kurti galimybę.

Induktyvinis metodas

Taikant induktyvinį metodą turimos žinios skiriamos į dvi grupes: bazinę teoriją (*angl. background theory*) B ir apibendrintinus duomenis E (be to, $B \not\models E$). Apibendrintini duomenys dar skirstomi į teigiamus E^+ ir neigiamus E^- pavyzdžius. $B \wedge E^- \not\models \perp$. Iškelta hipotezė H laikoma induktyviąja išvada, jeigu: (1) ji neseka iš turimų žinių, tačiau paaiškina teigiamus pavyzdžius: $B \wedge H \models E^+$; ir (2) neprieštarauja neigiamiems: $B \wedge H \wedge E^- \not\models \perp$. Egzistuoja bendrieji algoritmai hipotezėms generuoti ir joms tikrinti [14].

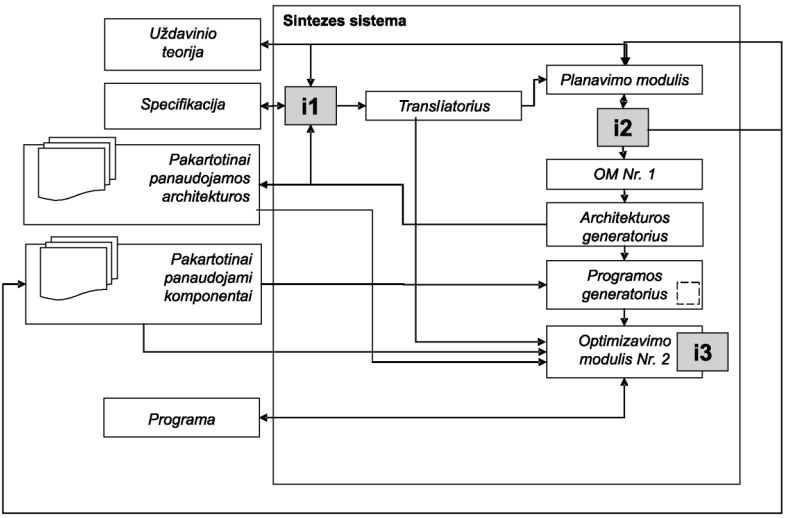
Pagrindinės induktyviojo loginio programavimo naudojimo sritys: vaistų gamyba, palydovinė orų prognozė, automatinis programavimas. Induktyvusis loginis programavimas naudojamas funkcinių programų (dažniausiai LISP kalba) kūrimui [1, 14], tačiau pastaruoju metu daugėja jo taikymų struktūrinei, objektinei ir aspektinei paradigmai [2, 3, 15].

Vienas induktyvinio metodo privalumų, lyginant jį su dedukciniu metodu – galimybė atlikti loginių programų sintezę remiantis ir nebaigtomis specifikacijomis [8, 14]. Šią metodo savybę galima panaudoti ir komponentinių programų specifikacijos problemai spręsti. Sintezės sistema papildoma moduliu ***i1***, kuris indukcijos būdu patikslintų nebaigtas ar nekorektiškas specifikacijas. Patikslintą specifikaciją toks elementas perduotų transliatoriui (1 pav.).

Kaip teigiama [14], induktyvinis metodas padeda nustatyti ryšius tarp konceptų. Pastaroji metodo savybė yra labai svarbi sprendžiant neapibrėžtųjų komponentų problemą.

Sintezės sistemoje tarp teoremos įrodymo planavimo modulio ir pirmojo optimizavimo modulio tikslinga įterpti dar vieną modulį ***i2*** (1 pav.). Jei planavimo modulis randa teoremos įrodymą, vadinasi uždavinys išspręstas ir neapibrėžtųjų komponentų problemos nėra. Šiuo atveju papildomas modulis neatliktų jokių papildomų veiksmų, o tiesiog perduotų planavimo modulio rezultatus architektūros generatoriui. Tačiau, jei teoremos įrodyti nepavyko, papildomas modulis, naudodamas indukcijos metodą, bando nustatyti, kokių aksiomų trūksta ir ar jos gali būti realizuotos. Generavęs papildomas aksiomas šis modulis gražina papildytą uždavinį planavimo moduliui.

Tam, kad būtų išspręsta nefunkcinių reikalavimų problema, sintezės sistema turi turėti galimybę įvertinti komponentus nefunkcinių reikalavimų atžvilgiu, t.y. sistema turi gebėti iš komponentų aibės parinkti tokį komponentą, kurio veikimo sparta, saugumo lygmuo ir kitos nefunkcinės savybės geriausiai atitinka specifikaciją. Komponentinei programų sintezės sistemai yra būtinas antrasis optimizavimo modulis ***i3***,



1 pav. Komponentinių programų sintezės sistema naudojanti induktyvinį metodą.

kuris ir atliktų šį uždavinį [7]. Induktyvinis metodas įgalina nustatyti komponentų alternatyvas sistemos nefunkciniams reikalavimams patenkinti. Be to, induktyvinis metodas yra naudojamas ieškant aukštesnio lygmens dėsningumams (*high-level regularitie*s) nustatyti [14], o tai gali padėti ieškant bendresnių komponentų ar jų interfeisų.

Įdėją panaudoti loginį programavimą (taip pat ir induktyvųjį loginį programavimą) komponentinėms programų sistemoms kurti taip pat plėtoja mokslininkų K.-K. Lau [9, 10, 11] ir M. Martelli [12] vadovaujamos tyrimų grupės.

Nepaisant plačių induktyviojo metodo galimybių, būtina atkreipti dėmesį, kad jis turi ir trūkumų:

- Kaip pabrėžia S. Muggleton [13], metodas gali būti neefektyvus tuo atveju, kai remiamasi vien tik pavyzdžiais. Pavyzdžių aibei sudaryti gali būti sunaudojama daugiau resursų (pvz.: laiko) daug daugiau nei programai kūrti.
- Nėra garantijų, kad indukcijos rezultatas visada bus korektiškas [1, 14]. Struktūrinė programų sintezė ir kiti deduktyviniai metodai užtikrina, kad gauta išvada yra teisinga. Programų sintezės sistemos, naudojančios deduktyvinius metodus rezultatas taip pat yra visada korektiška. Tuo tarpu taikant induktyvųjį metodą remiantis tais pačiais pavyzdžiais gali būti gautos skirtingos išvados.

Išvados

Induktyvinis metodas komponentinių programų sistemų sintezės sistemoje gali būti sėkmingai naudojimas kartu su struktūrinės sintezės metodu siekiant išspręsti šias problemas:

- specifikacijos problemą;
- neapibrėžtųjų komponentų problemą;
- nefunkcinių reikalavimų problemą.

Tačiau, siekiant išvengti nekorektiško rezultato, visos sintezės metu padarytos induktyviosios išvados privalo būti papildomai verifikuojamos.

Literatūra

1. Y.M. Barzdin', A.N. Brazma, E.B. Kinber, Inductive synthesis of programs: State of the art, problems, prospects, *Cybernetics and Systems Analysis*, **23**(6), 818–826 (1987), http://dx.doi.org/10.1007/BF01070244.
2. D. Binkley, M. Ceccato, M. Harman, F. Ricca, P. Tonella, *Automated Refactoring of Object Oriented Code into Aspects* (2005).
3. S. Ferilli, F. Esposito, T.M.A. Basile, N. Di Mauro, Automatic induction of first-order logic descriptors type domains from observations, *Lecture Notes in Computer Science*, **3194**, 116–131 (2004).
4. V. Giedrimas, A. Lupeikienė, Komponento specifikacijos formalizavimas, *Liet. matem. rink.*, **44**(spec. nr.), 276–280 (2004).
5. V. Giedrimas, A. Lupeikienė, Komponentinių programų struktūrinės sintezės teorinės problemos, *Liet. matem. rink.*, **45**(spec. nr.), 139–143 (2005).
6. V. Giedrimas, Component-based software generation: the structural synthesis approach, in: *Proceedings of 7th GPCE YRW 2005*, Tallinn (2005), pp. 19–23.
7. V. Giedrimas, Architectures of component-based structural synthesis systems, in: *Proceedings of Baltic DB & ISÆ06* (2006), pp. 311–315.
8. A. Guessoum, J. Komorowski, *Induction, Deduction and Abduction for Program Design and Maintenance* (1995).
9. K.-K. Lau, A. Momigliano, M. Ornaghi, Constructive specification of compositional units, in: *Proceedings of the Fourteenth International Workshop on Logic-based Program Synthesis and Transformation*, *LNCS*, **3573**, Springer-Verlag, 198–214 (2005).
10. K.-K. Lau, Component-based software development and logic programming, in: *Proceedings of the Nineteenth International Conference on Logic Programming*, *LNCS*, **2926**, Springer-Verlag, 103–108 (2003).
11. K.-K. Lau, Component-based software development and logic programming, *LNCS*, **2916**, 103–108 (2003).
12. M. Martelli, V. Mascardi, F.Zini, A logic programming framework for component-based software prototyping, in: *Proceeding of 2nd International Workshop on Component-based Software Development in Computational Logic* (*COCL' 99*) (1999).
13. S. Muggleton, C. Feng, Efficient induction of logic programs, in: *Proceedings of the 1st Conference on Algorithmic Learning Theory* (1990), pp. 368–381.
14. S. Muggleton and L. De Raedt, Inductive logic programming: Theory and methods, *Journal of Logic Programming*, **19**(20), 629–679 (1994).
15. T. Tourwé, J. Brichau, A. Kellens, K. Gybels, Induced intentional software views, *Computer Languages, Systems & Structures*, **30**(1–2), 35–47 (2004).
16. E. Tyugu, *Knowledge Based Programming*, Turing Institute Press (1988).

SUMMARY

V. Giedrimas. Induction in component-based software synthesis

The automatic programming and automatic software synthesis systems are relevant software engineering research objects. This article presents the prospects of induction as a component-based software synthesis method. The usage of induction in the component-based software synthesis system can solve main synthesis problems: problem of specification, problem of undefined components and problem of non-functional requirements. However using induction synthesis system can lose soundness provided by Structural synhesis of programs method. To avoid this any inductive conclusion should be verified.

Keywords: component-based system, software synthesis, induction.