نقش و کاربرد گرامرهای ویژه در مهندسی معکوس نرم افزار

تمرین (۲) درس کامپایلر پیشرفته نیمسال ۱-۱۳۹۵

دانشجو:

مرتضی ذاکری

استاد:

دکتر سعید پارسا

پاییز ۱۳۹۵

۱ مقدمه

در این نوشـــتار به صـــورت خلاصـــه به کاربردها و نحوه اســـتفاده از *گرامرهای ویژه*[۱] (گرامرهای دارای صـــفت (رجوع شـــود به مبحث کامپایلرها)، در مباحث مربوط به *مهندســـی معکوس نرم افزار*[۲] می پردازیم. ابتدا مفاهیم گرامرهای ویژه و مهندســـی معکوس را شـــرح می دهیم و سپس چکیده چند مقاله و گزارش کاربردی مرتبط در این زمینه را مرور می کنیم.

۲ مفاهیم پایه

۲ـ۱ گرامر ویژه

طبق کتاب [1] در تعریف گرامر ویژه هر قانون تولید گرامر $A \to \alpha$ متناظر با یک مجموعه از قوانین معنایی به صورت $b := f(c_1.c_2.\ldots.c_k)$ که در آن f یک تابع است و b:

۱- b یک صفت اکتسابی از A و $c_1.c_2.\ldots.c_k$ صفت هایی هستند که متعلق به نماد های گرامر قانون تولید هستند و یا

۲- b یک صـــفت موروثی از یکی از نمادهای گرامر در طرف راســـت قانون تولید اســـت و $c_1.c_2.\ldots.c_k$ صفت هایی متعلق به نمادهای گرامر هستند.

پایانه ها بنا به فرض فقط دارای صفت های اکتسابی هستند، که معمولا توسط تحلیل گر لغوی فراهم می شود. غیر پایانه ها دارای صفت های موروثی نیز هستند که در درخت تجزیه عبارت ظاهر می شود.

دو طرح و اســـتفاده غالب برای گرامر ویژه در کامپایلر مطرح می گردد. یکی در ایجاد صفات مربوط به آزمون نوع در فاز تحلیل معنایی برنامه و دیگری ایجاد صفات لازم برای تولید کد در مرحله تحلیل نحوی (ترجمه هدایت شـــده یا مبتنی بر دســـتور). البته این فاز ها معمولا همزمان انجام می پذیرد. به عبارت ســـاده در گرامرهای ویژه برای هر غیرپایانه یا متغیر میانی یک صـــفت از جنس رشـــته در نظر گرفته می شـــود که محتوای آن حاوی **کدی** اســـت که برای متغیر میانی ایجاد می شود.

---

[۱] Attributed Grammars

[۲] Software Reverse Engineering

۲ـ۲ مهندسی معکوس

واژه مهندسی معکوس نخستین بار از در فناوری های سخت افزاری ظاهر شد و به فرایند اقتباس ویژگی ها و ساختار یه سخت افزار پیچیده اشاره می کند. اخیرا این اصطلاح در حوزه نرم افزار نیز وارد شده است که اگر چه معنایی مشابه دارد اما هنوز تعریف دقیقی از آن ارائه نشده است. طبق [2] مهندسی معکوس عبارت است از:

> *«فرایند تحلیل یک سیستم به منظور شناسایی اجزای سیستم و روابط داخلی حاکم بر آن ها، و ایجاد یک نمایش از سیستم در شکل دیگر و سطح بالاتری از انتزاع.»*

۳ کاربردها

۳ـ۱ آیا گرامرهای ویژه در صنعت کاربرد دارد؟

گرامرهای ویژه هم در مطالعه های آکادمیک و هم در کارهای صنعتی کاربرد دارند. اما به جنبه دوم توجه کمتری شده است زیرا اغلب افراد درحوزه کامپیوتر از آن بی اطلاع هستند. اجازه دهید ببینیم این ساختار ها کجا و چگونه کاربرد دارند؟

یکی از کمپانی هایی که در زمینه استفاده از این گرامرها فعالیت دارد کمپانی آلمانی **CoCoLab** است[۳]. مهمترین زمینه تجارتی این شرکت پارسرها یا front-end ها برای زبان هایی مثل COBOL، PL/1، C، C++ و جاوا است. این پارسرها معمولا برنامه های تحلیل گر در پروژه های مهند سی معکوس کاربرد دارند. در این پار سرها گرامرهای ویژه برای دو وظیفه به کار می روند. نخست، ساخت درخت تجزیه و دیگر درخت ها در حین عملیات تجزیه. دوم، برای تحلیل نام در تحلیل مفهومی که بر روی درخت نحوی مجازی انجام می شود. توسعه گرامرهای ویژه برای زبان های مختلف زمان بر است و بین یک روز تا یک یا دو ماه طول می کشد. این زمان بستگی به اندازه و پیچیدگی گرامر دارد. برای مثال گرامر زبان COBOL تقریبا ۳۰۰۰ قانون دارد. با این حال این عمل یک بار انجام می شود و سپس قابل استفاده روی تمام کدهای نوشته شده به آن زبان خواهد بود [3].

[۳] www.cocolab.de

```
postfix_expression = <
        = primary_expression .
        = p:postfix_expression '[' e:expression ']'
        { type := type_op (p:type, karray);
          tree := msubscript_expr ('[':Position, p:tree, e:tree); } .
        = postfix_expression '(' expression_list ')'
        { type := type_op (postfix_expression:type, kfunction);
          tree := mcall_expr ('(':Position, postfix_expression:tree,
                ReverseTree (expression_list:tree)); } .
        = s:simple_type_specifier '(' expression_list ')'
        { type := get_type (s:tree);
          tree := (s:tree->specifier.next = mnospecifier (),
                mconstruct_expr ('(':Position, s:tree,
                ReverseTree (expression_list:tree))); } .
        = ...
```

شکل ۱ نمونه ای از گرامر ویژه ساخت درخت تجزیه نحوی در C++

گرامر ویژه فوق یک نمونه کاربرد کوچک در روی گرامر زبان C++ را نشـــان می دهد. دو صفت در قواعد این گرامر محاسبه می شوند. صفت tree که برای ساخت درخت تحلیل نحوی به کار می رود. صفت type که نوع هر عبارت را محاسبه می کند.

۲ـ۳ ابزارهای خاص کمپانی CoCoLab

برخی دیگر از کارکردهایی که این کمپانی با ا ستفاده از گرامرهای ویژه ایجاد کرده ا ست عبارتند از:

- **انتقال درخت در زبان PL/1**[۴]: در این کاربرد که مخ صوص مهند سی معکوس کد در زبان PL/1 هست، تشخصی فراخوانی تابع به صورت F(X) حل شده است. در این زبان F(X) ب سته به تعریف F هم ن شان دهنده فراخوانی تابع ا ست و هم ن شان دهنده د ستر سی به عنا صر آرایه. با توجه به این که PL/1 اجازه ا ستفاده قبل از تعریف را می دهد، تحلیل گر نحوی ممکن اســـت تعریف F را هنگام ســـاخت درخت تجزیه متوجه نشـــود. بنابراین، پارســـر این متغیر را به یک گره درخت در اولین مکان ممکن نگاشـــت می کند. بعدا یک انتقال درخت تجزیه صورت می گیرد و نتایج تحلیل مفهومی م شخص می کند که نوع کاربرد متغیر

[۴] Tree Transformation in PL/I

نام برده نهایتا چیست. این انتقال به صورت گرامرهای ویژه و با استفاده از یک برنامه برنامه پردازش گر درخت به نام puma نوشته شده است.

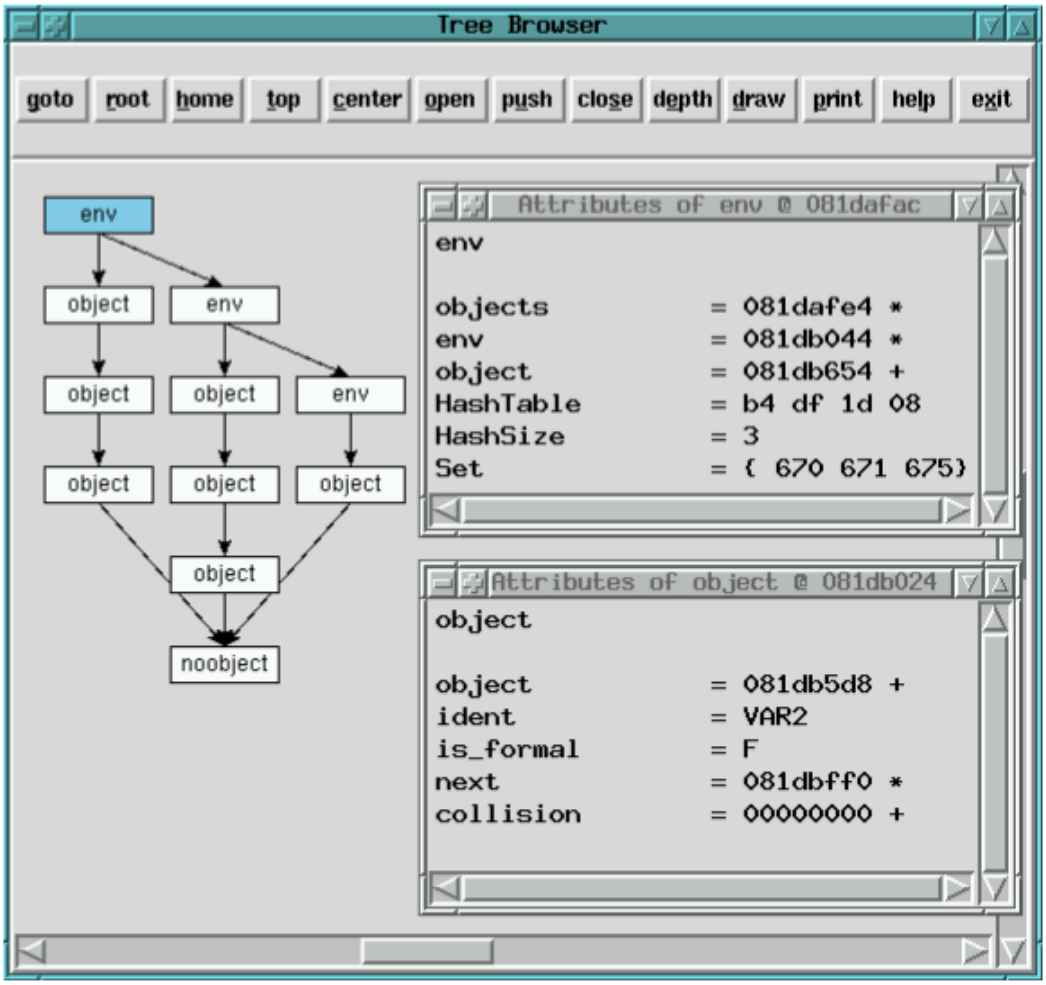
- **اعتبار سنجی اسناد XML**[۵]: با استفاده از گرامر های DTD که نوع محدود تری از گرامرهای آزاد یا مستقل از متن است. می توان یک سند XML را اعتبارسنجی کرد. این عمل به طبقه بندی اسناد xml کمک می کند. مشابه مورد قبلی این کار به وسیله ترکیب استفاده از گرامر های ویژه و پردازش گر درخت puma قابل انجام است.
- **الگوریتم نمایه**[۶]: در این کارکرد ابزار مدیریت درخت ast از بسته نرم افزاری Cocktail یک شمای گرافیکی از گراف ها و درخت ها که از کد برنامه استخراج شده اند را نشان می دهد. گرامر ویژه نمایش داده شده در زیر بخشی از گرامر لازم برای انجام این کار است.

[۵] Validation of XML Documents

[۶] Layout Algorithm

```
RULE
root    = tree .
tree    = <
   node = child1:tree   child2:tree child3:tree .   /* node with 3 children */
   list = next:tree REV child2:tree child3:tree .   /* list node */
   leaf = .                                         /* leaf node */
> .
MODULE compute_layout
DECLARE tree = [xin INHERITED]   [yin INHERITED]
               [x   OUTPUT]      [y   OUTPUT]
               [w   SYNTHESIZED] [h   SYNTHESIZED] .   /* width and height */
RULE
root    = { tree:xin := 0; tree:yin := 0; } .
tree    = { x := xin; y := yin; w := 0; h := 0; } .
leaf    = { x := xin; y := yin; w := 1; h := 1; } .
node    = { x := (child1:x + child3:x) / 2; y := yin;
            w := child1:w + child2:w + child3:w;
            h := Max (Max (child1:h, child2:h), child3:h) + 1;
            child1:yin := yin + 1; child1:xin := xin;
            child2:yin := yin + 1; child2:xin := xin + child1:w;
            child3:yin := yin + 1; child3:xin := xin + child1:w + child2:w;
          } .
list    = { x := xin; y := yin;
            w := Max (1 + child2:w + child3:w, next:w);
            h := Max (child2:h, child3:h) + next:h;
            next:yin := yin + Max (child2:h, child3:h); next:xin := xin;
            child2:yin := yin + 1; child2:xin := xin + 1;
            child3:yin := yin + 1; child3:xin := xin + 1 + child2:w;
          } .
END compute_layout
```

شکل ۲ گرامر ایجاد گرافیکی درخت تجزیه



شکل ۳ درخت تجزیه حاصل از مهندسی معکوس کد

۳ـ۳ مهندسی معکوس مبتنی بر گرامر برای تایید رفتار و تست نرم افزار

در مقاله [4] رو شی با ا ستفاده از گرامر های گراف برای برر سی و تایید رفتار سیستم های با بی درنگ یا تضمین شده[۷] شرح داده شده ا ست. در این مقاله رو شی شرح داده شده است که با استفاده از گرامر، متن برنامه را به صورتی که زبان های ویژوال و نمودار های معمول UML نشان می دهند، بازنمایی می کند. به این ترتیب نحوه برر سی رفتار چنین سیستمی در مدل جدید بسیار آسان تر خواهد بود.

[۷] High Assurance System

۴ نتیجه گیری

با توجه به موارد گفته شـــده می توان گفت امروزه ضـــرورت اســـتفاده از تکنیک های موجود مباحث مربوط به کامپایلر، در مهندســی معکوس نرم افزار کاملا محســوس اســت. با این حال وسعت کاربرد بسیاری از این تکنیک ها به دلیل عدم وجود ابزارهای قوی و شناخته شده، همچنان اندک اســـت و بیش تر توجه های صـــورت گرفته در حوزه آکادمیک هســـتند تا حوزه تجاری. گرامرهای ویژه نیز در این میان نقش منحصـــر به فردی داشـــته و به کمک آن می توان تقریبا هر سیستم نرم افزاری را که کد قابل دستر سی دارد، مهند سی معکوس نمود. تو سعه ابزارهای مهندسی معکوس مبنی بر گرامرهای ویژه مبحثی آکادمیک و در عین حال کاربردی و جذاب خواهد بود.

۵ منابع

[1] V. Aho, R. Sethi and D. Ullman, Compilers: Principles, Techniques and Tools, US: Pearson, 2014.

[2] M. v. d. Brand , P. Klint and C. Verhoef, "Re-engineering needs Generic Programming Language Technology," p. 17, 1996.

[3] J. Grosch, "Are Attribute Grammars Used in Industry?," CoCoLab - Datenverarbeitung, Germany.

[4] C. Zhao and K. Zhang , "A Grammar-Based Reverse Engineering Framework for Behavior Verification".