تحلیل و طراحی کامپایلر سیستم-L

علیرضا حجه‌فروش
ar.hajehforoush@gmail.com

مهدی وحیدی پورعضو هیئت علمی دانشگاه، دانشگاه کاشان،
دانشکده مهندسی، گروه کامپیوتر
Vahidipour@kashanu.ac.ir .

چکیده

در سال 1968 یک زیست شناس نوع جدیدی از مکانیزم بازنویسی رشته‌ها را معرفی کرد که در آینده سیستم- L نام گرفت. دوباره-نویسی تکنیکی برای تعریف شی‌های پیچیده است. در این روش با جایگزین‌کردن متوالی قسمتی از شکل اولیه بوسیله قانون‌های تولیدی، دوباره نویسی انجام می‌شود. در سیستم-L اعمال قوانین بر خلاف سیستم گرامر چامسکی به صورت موازی انجام می‌شود. انواع مختلفی از سیستم-L مانند براکت‌دار، احتمالی، پارامتری و مخصوص مدل گیاهان تعریف شده است. در تمامی آن‌ها بعد از آن‌که قوانین تولیدی، رشته حاصل از بازنویسی را تولید نمود شکل گرافیکی متناظر از تفسیر هندسی آن قابل ارائه است. لذا در این مقاله، با در نظر گرفتن قسمت تولید رشته کامپایلری طراحی شده است که می-تواند برنامه‌ای با زبان سیستم-L و انواع مختلف آن‌را تحلیل نموده و اجرا نماید.

کلمات کلیدی

سیستم-L - تفسیر لاک پشتی - گرامرهای بازگشتی - کامپایلر - تحلیل‌گر - قوانین تولید

1- مقدمه

سیستم Lindenmayer یا به اختصار سیستم-L برای ایجاد گیاهان براساس تئوری‌های ریاضی قابل اجرا می‌باشند [1]. مفهوم اصلی سیستم-L دوباره‌نویسی می‌باشد. به طور اساسی، دوباره‌نویسی یک تکنیک برای تعریف شی‌های پیچیده می‌باشد که با جایگزین‌کردن متوالی قسمتی از شی اولیه با استفاده از قانون‌های تولیدی دوباره نویسی انجام می‌شود. [2].

پس از آن‌که سیستم-L با استفاده از بازنویسی، رشته‌ای را تولید می‌کند نوبت به تفسیر هندسی آن می‌رسد. تفسیر لاک‌پشتی[1] یکی از رایج‌ترین آنهاست[1].

در این مقاله سعی بر آن است که کامپایلری طراحی شود که بتواند برنامه‌ای با زبان سیستم-L و انواع مختلف آن‌را تحلیل نموده و رشته حاصل از آن‌را برای مفسرهای مختلف هندسی آماده نماید. در ابتدا فرم کلی سیستم-L در قسمت 2 بررسی می‌شود و سپس انواع در نظر گرفته شده‌ی آن برای طراحی کامپایلر بیان می‌شود. در قسمت 3 قسمت‌های مختلف کامپایلر سیستم-L بیان شده است.

2- مفاهیم سیستم-L

تفاوت اصلی که میان گرامر شامسکی[2] و سیستم- L وجود دارد روش اعمال قوانین آن است. در گرامرهای شامسکی قوانین به صورت تدریجی اعمال می‌شوند، در حالیکه در سیستم-L آن‌ها به صورت موازی اعمال می‌شوند و همزمان همه‌ی علائم با هم در کلمه جایگزین می‌شوند.

2-1- انواع سیستم-L

برای در برگرفتن ساختار بیشتری از طراحی گرافیکی نوع‌هایی برای سیستم- L در نظر گرفته شده تا طراحی‌ها هر چه بیشتر به حالت طبیعی نزدیکتر باشد. [3].

سیستم-L براکت دار[3] برای شاخه‌ها استفاده می‌شود. ابتدای شاخه با '[' و انتهای آن با ']' مشخص می‌شود. براکت باز به معنای قراردادن حالت لاک-پشت درون استک و براکت بسته به معنای برداشتن حالت از روی استک می‌باشد[2,4]. در یک سیستم- L احتمالی[4] هر قانون با یک احتمال همراه است که نشان‌دهنده احتمال اعمال آن است [5,6]. در سیستم-L حساس به متن[5] شناسه‌ی قابل جایگزینی دارای شرایطی برای متن سمت راست و چپ خود است. سیستم-1L برای جایگزینی شناسه تنها به یک سمت آن توجه دارد. اما در سیستم- 2L دو طرف راست و چپ شناسه‌ی قابل جایگزینی باید شامل متن‌های در نظر گرفته شده باشد تا قانون اعمال شود[7,8,9]. اگر در داخل سیستم برای هر شناسه پارامتری تعریف می‌شود که در صورت برآورده کردن شرط روی آن پارامتر، آن قانون اعمال می‌شود سیستم- L پارامتری[6] تعریف شده است. در یک سیستم بنام سیستم- L مدل کردن اندام گیاهان از نشانه‌های '.'، '{' و '}' برای رسم برگ‌های گیاهان استفاده می‌شود. علامت نقطه نشان‌دهنده‌ی راس‌های چندضلعی می‌باشد. علامت '{' شروع رسم چندضلعی و علامت '}' نشان دهنده‌ی متصل کردن نقاط مشخص شده می‌باشد[10].

3- طراحی کامپایلر

قبل از طراحی کامپایلر[11,12,13] سیستم- L اجزا و ساختاری که یک نوشتار در این سیستم باید دارا باشد بیان می‌شود: سیستم-L بازنویسی خود را از قاعده‌ی اصلی یا پایه[7] شروع می‌نماید. قوانین دیگر بر روی آن اعمال می‌شوند و جمله‌ی نهایی را تولید می‌کند. در نتیجه می‌توان اجزای سیستم-L را به سه دسته‌ی زیر تقسیم کرد: 1) الفبای زبان: در این طراحی فرض شده است که الفبای زبان در داخل یک جفت براکت باز بسته

به کامپایلر معرفی می‌شوند. هر یک از الفبا از دیگری با کاما جدا می‌شود مانند:[a , b2]. 2)-قوانین تولید: فرمت نوشتن آن باید طوری بیان شود که تمام نوع‌های سیستم-L مطرح شده را در بر گیرد. ساختار در نظر گرفته شده در این مقاله به صورت شکل (1) می‌باشد. حروفی که زیر آنها خط کشیده شده است کلمات کلیدی می‌باشند و باید عیناً در قوانین ذکر شوند. شناسه‌ی Letter معرف سمت چپ قانون می‌باشد و عمل جایگزینی روی آن انجام می‌شود. پارامتر *t*، در سیستم‌های-L پارامتری در شرط test استفاده می‌شود. اگر محتوای سمت چپ و راست *Letter* بترتیب *Left context* و*Right context* باشند این قانون جایگزینی انجام می‌شود. مقدار محتوی چپ و راست می‌تواند به سه طریق بیان شود: با یک شناسه ، فضای خالی و '*' . اگر قبل ازیک شناسه '!' گذاشته شود به این معنی است که این شناسه در محاسبه‌ی متن چپ و راست متن همسایه نادیده گرفته شود. شرط *Test* به صورت پارامتری می‌باشد. برای بیان طرف راست قانون *Last production* در نظر گرفته شده‌است که بعد از "<>" نوشته می‌شود. در متن سمت راست، هر شناسه می‌تواند با یک عدد و یا پارامتر همراه شود. در سیستم‌های-L احتمالی از "<>" استفاده می‌شود که داخل آن مقدار احتمال اعمال این قانون نوشته می‌شود. 3) قاعده‌ی پایه: قاعده‌ی اصلی مانند طرف چپ یک قانون تولید نوشته می‌شود با این تفاوت که پارامترها فقط باید عدد باشند یعنی در پارامترها نمی‌توانیم *t* یا یک عملگر را داشته باشیم.

3-1- تحلیلگر لغوی

نخستین مرحله کامپایل تحلیل واژه‌ای یا لغوی می‌باشد و کار آن تشخیص توکن‌ها می‌باشد.توکن‌های مجاز در سیستم- L عبارتند از:

id Num < = > * + - / ^ (
!) { . } [ ] ; : ~ t " ,

اگر نشانه‌ای به غیر از نشانه‌های بالا در ورودی باشد خطایی از تحلیل‌گر لغوی صادر می‌شود . همچنین فضاهای خالی و توضیحات بوسیله‌ی تحلیل‌گر لغوی از بین می‌رود. برای یافتن نقش توکن '<' و '>' نمی‌توان از تحلیل‌گر لغوی استفاده نمود، به‌همین دلیل از تحلیل‌گر نحوی استفاده می‌کنیم. همچنین چون '*' فقط نقش ضرب را ندارد و می‌تواند در *Left Context* و *Right Context* هم به کار رود نقش آن در تحلیل‌گر نحوی معلوم می‌شود.

برای پیاده سازی اسکنر یک زبان ابتدا باید فرم کلی توکن‌های آن زبان را توسط عبارات منظم توصیف کرد. در اینجا توکن *id* بسته به نوع الفبا تعریف می‌شود (الفبا شامل حروف و اعداد می‌باشد). توکن *num* بسته به نوع عددهایی که می‌توانیم داشته باشیم توصیف می‌شود (اعداد اعشاری، صحیح). نکته‌ای که باید توجه کرد توکن *t* می‌باشد که با توجه به نشانه‌ی خوانده شده‌ی بعدی می‌توان آن را از توکن *id* تمییز داد، زیرا بعد از توکن *id* نشانه‌های ')' یا ']' یا ',' می‌آید در صورتی‌که برای *t* این‌طور نمی‌باشد.

3-2- تحلیلگر نحوی

بعد از نوشتن گرامرهای تولید سیستم-L باید تحلیل نحوی انجام شود(جداول 1و2). به‌طور کلی سه نوع روش تحلیل نحوی وجود دارد[11] : روش عمومی که چندان کارا نیست ولی با هر نوع گرامری کار می‌کند، روش‌های بالابه‌پایین و روش‌های پایین‌به‌بالا. در این مقاله از روش بالابه-پایین استفاده شده است. روش‌های پایینگرد و LL(1) مهمترین روش‌های تجزیه بالابه‌پایین هستند. روش LL(1) که حالت خاصی از روش کلی LL(K) می‌باشد در این طراحی استفاده‌شده است. خصوصیاتی که گرامرهای LL(1) باید دارا باشند را در گرامرهای تولیدی لحاظ می‌کنیم. با بدست آوردن توابع First و Follow برای گرامرها جدول LL(1) آن را رسم می‌کنیم.

3-2-1- روش‌های اصلاح خطای نحوی در روش تجزیه LL(1)

برای اصلاح خطا از روش Panic Mode استفاده می‌کنیم. یک انتخاب مناسب در نظر گرفتن مجموعه Follow هر غیرپایانه‌ای به عنوان مجموعه‌ی Synchronizing آن غیرپایانه است. برای اصلاح خطا به روش Panic Mode در الگوریتم تجزیه LL(1) بصورت زیر عمل می‌کنیم:

- اگر پارسر خانه M[A,a] راخالی ببیند، علامت a را در ورودی نادیده می‌گیرد.
- اگر در محل خانه‌ی M[A,a] علامت "S" باشد غیرپایانه‌ی بالای انباره حذف می‌شود، مشروط بر آن که A تنها غیرپایانه موجود در انباره نباشد.
- اگر پایانه‌ی انباره با ورودی جاری تطبیق نکند، پایانه‌ی بالای انباره حذف می‌شود.

3-3- تحلیلگر معنایی

پس از خاتمه مراحل تحلیل‌گر لغوی و نحوی ، کامپایلر برنامه ورودی را از نظر معنا مورد بررسی قرار می‌دهد . عملیاتی که در تحلیل معنایی در کامپایلر سیستم-L انجام می‌شود کنترل بودن شناسه‌ها جزء الفبا می‌باشد.

3-4- تولید کد

قبل از تولید کد باید ببینم برای اعمال قوانین احتیاج به ذخیره‌ی چه مقادیری هنگام کامپایل رشته‌های سیستم-L داریم. به همین خاطر باز به اجزای سیستم-L برمی‌گردیم و هر یک را جداگانه بررسی می‌کنیم. اگر سیستم-L را یک کلاس کلی در نظر بگیریم این کلاس شامل سه زیر کلاس الفبا (*Alphabet*)، قوانین تولید (*ListProduction*) و قاعده‌ی پایه(*State*) می‌باشد. درکلاس الفبای زبان هر *id* باید به عنوان الفبای به کار رفته در آن سیستم-L ذخیره شود تا هنگام مشاهده‌ی *id* در قوانین، کنترل شود که جزء الفبا باشد. در کلاس *ListProduction* به ازای هر سیستم-L نیاز به یک یا چند قانون(*Production*) داریم. همانطور که بیان شد یک قانون تولید از دو بخش تشکیل می‌شود. بخش اول که شرایط اعمال بازنویسی قانون با شناسه را بیان می‌کند

"letter(t) _ left context _ right context _ [test] :

و دنباله‌ی قانون که در صورت برقراری شرط، جایگزین *Letter* می‌شود
production list < > last production"

"letter(t) , left context , right context , [test] : production list < > last production"

شکل 1- ساختار در نظر گرفته شده برای قوانین

جدول (1): گرامرهای الفبا و قوانین تولید

1	START → FUNCTION;START	30	K → ! L	59	RCONTEXT → ε
2	START → ALPHABET	31	K → [K] K	60	RULE → < POSSIBLE
3	FUNCTION → ~FID(t) : ALFA	32	K → L	61	POSSIBLE → DIGIT X , RULE
4	ALFA → ε	33	K → . K	62	POSSIBLE → X
5	ALFA → [F	34	K → { K } K	63	X → > APPLY
6	F → AID G	35	K → ε	64	APPLY → ! U
7	G →] PRODUCTION	36	L → id (N) K	65	APPLY → [APPLY] APPLY
8	G → , F	37	N → t Q	66	APPLY → U
9	PRODUCTION → " SUBPRODUCTION	38	N → NUM1	67	APPLY → . APPLY
10	PRODUCTION → ε	39	Q → OPCODE NUM1	68	APPLY → { APPLY } APPLY
11	SUBPRODUCTION → P " E	40	Q → ε	69	APPLY → ε
12	E → PRODUCTION	41	ALFABET → [R	70	U → id (N) APPLY
13	E → ε	42	ALFABET → ε	71	U → ~ id (N) APPLY
14	P → D(t),LFCONTEXT,RFCONTEXT,[B]:C	43	R → AID V	72	D → id
15	LFCONTEXT → *	44	V →] PROD	73	OPCODE → +
16	LFCONTEXT → id	45	V → , R	74	OPCODE → -
17	LFCONTEXT → ε	46	PROD → " SUBPROD	75	OPCODE → *
18	RFCONTEXT → *	47	PRODU → ε	76	OPCODE → /
19	RFCONTEXT → id	48	SUBPROD → W " Z	77	OPCODE → ^
20	RFCONTEXT → ε	49	Z → PROD	78	FID → id
21	B → t OP NUM	50	Z → ε	79	AID → id
22	B → ε	51	W → D(t),LCONTEXT,RCONTEXT,[B]: RULE	80	NUM → num
23	OP → <	52	LCONTEXT → *	81	NUM → + num
24	OP → =	53	LCONTEXT → id	82	NUM → - num
25	OP → >	54	LCONTEXT → ~ id	83	DIGIT → num
26	C → < I	55	LCONTEXT → ε	84	DIGIT → + num
27	I → DIGIT J , C	56	RCONTEXT → *	85	NUM1 → num
28	I → J	57	RCONTEXT → id	86	NUM1 → + num
29	J → > K	58	RCONTEXT → ~ id	87	NUM1 → - num

جدول (2):گرامرهای قاعده‌ی اصلی

1	START → S
2	S → ! B
3	S → [S] S
4	S → B
5	S → . S
6	S → { S } S
7	S → ε
8	B → id (T) S
9	B → ~ id (T) S
10	T → num
11	T → + num
12	T → - num

بخش اول در هر قانون واحد می‌باشد در صورتیکه طرف دوم برای سیستم-L احتمالی به ازای هر مقدار احتمال دارای یک خروجی برای طرف چپ خود می‌باشد .از بخش اول هر کلاسِ *Production* احتیاج به مقادیر زیر دارد:

- ذخیره کردن مقدار *letter*(input)،

- داشتن و یا نداشتن محتوای راست(HasRightContext)،
- داشتن و یا نداشتن محتوای چپ(HasLeftContext)،
- ذخیره‌ی محتوای راست(RightContext)،
- ذخیره‌ی محتوای چپ(LeftContext)،
- داشتن یانداشتن شرط(HasCondition)،
- مقدار شرط(ConditionValue)،
- نوع عملگر در شرط(ConditionOp)،
- ذخیره کردن مقدار احتمالات(OutputProbabilityTable) و
- شیئی از کلاس خروجی‌ها(ListOutput)

این‌ها مواردی است که ما در ابتدای یک قانون مشخص می‌کنیم اما به ازای این مقادیر اگر حالت احتمالی باشد این قانون به ازای هر مقدار احتمال یک خروجی(Output) برای طرف چپ خود دارد. پس هر ListOutput شامل یک یا چند خروجی(Output) می‌باشد و هر Output از طرف چپ دارای ویژگی های زیر می‌باشد:

- مشخص کردن کدی برای نمادهای کاربردی مانند براکت باز در هنگام برخورد با آن‌ها و ذخیره‌ی آن مقدار(SpecialCode)،
- ذخیره کردن مقدار شناسه(Name)،
- ذخیره کردن مقدار عددی(Val)،
- ذخیره کردن نوع عملگر(OpCode)،
- تعیین به کاربردن یا به کارنبردن!(InContext)

کلاس قاعده‌ی اصلی(State) مانند طرف دوم قوانین تولید فقط با تفاوت ذکر شده است و به ازای هر State، یک یا چند نشانه(StateEntry) داریم که هر StateEntry احتیاج به ویژگی‌های زیر دارد:

- مشخص کردن کدی برای نمادهای کاربردی مانند براکت باز(SpecialCode)،
- ذخیره کردن مقدار شناسه(Name)،
- ذخیره کردن مقدار عددی(Val)،
- تعیین به کاربردن یا به کارنبردن!(InContext)

حال با توجه به کارهایی که هنگام کامپایل باید انجام شود علائم کنش را به قواعد گرامر کامپایلر اضافه می‌کنیم که از آن‌ها در تولید کد بالابه-پایین جهت هدایت عمل تولید کد استفاده می‌گردد. این علائم بایستی در محل‌های مناسبی در سمت برخی از قواعد گرامر قرار داده شوند. برای آنکه علائم کنش از سایر علائم گرامر (پایانه‌ها و غیر پایانه‌ها) تفکیک شوند ، معمولاً یک کاراکتر ویژه(مثلاً #) در جلوی آن قرار داده می‌شود. در طی تجزیه بالابه‌پایین ، هر زمان که یک علامت کنش بالای انباره پارس قرار گیرد، پارسر برنامه تولید کد را فرامی‌خواند و آن علامت کنش را به عنوان یک پارامتر به برنامه تولید کد ارسال می‌کند ، برنامه تولید کد نیز با استفاده از علامت کنش دریافت شده ، روال مفهومی مربوطه را پیدا و اجرا می‌کند. در روال‌های مفهومی مربوطه عمل ذخیره‌ی ویژگی را با توجه به مکان آن انجام می‌دهیم.

مانند گرامر $LCONTEXT \rightarrow id$ که با علائم کنش به گرامر زیر تبدیل می‌شود:

$LCONTEXT \rightarrow \#IDLCONTEXT\ id$

و روال مفهومی #IDLCONTEXT برابر است با:

در صورت بودن id جزء الفبای زبان

hasLeftContext =true , LeftContext=id

4- نتیجه‌گیری

در این مقاله، سعی بر آن شد که معرفی اجمالی از سیستم-L انجام شود و با توجه به راهبرد آن در گرافیگ، کامپایلری برای آن معرفی شود. همان-گونه که اشاره‌شد می‌توان تولید اشکال از یک برنامه در سیستم-L را به دو قسمت تولید رشته حاصل از بازنویسی و تفسیر گرافیکی آن رشته تقسیم نمود. این مقاله با در نظر گرفتن قسمت اول کامپایلری طراحی نموده است که می‌تواند برنامه‌ای با زبان سیستم-L و انواع مختلف آن‌را تحلیل نموده و اجرا نماید.

مراجع

[1] A. Lindenmayer. Mathematical models for cellular interaction in development, Parts I and II. Journal of Theoretical Biology,18:280–315, 1968.
[2] P. Prusinkiewicz, Graphical applications of L-systems, In Proceedings of Graphics Interface, Vision Interface 86, P: 247–253, CIPS, 1986.
[3] Przemyslaw Prusinkiewicz With James S. Hanan,F. David Fracchia,Martin J. M. de Boer, The Algorithmic Beauty of Plants,2004
[4] P. Prusinkiewicz and etal, Applications of L-systems to computer imagery, In H. Ehrig, M. Nagl, A. Rosenfeld, and G. Rozenberg, editors, Graph grammars and their application to computer science, Third International Workshop, Lecture Notes in Computer Science 291,P: 534–548, Springer-Verlag, Berlin,1987.
[5] P. Eichhorst and W. J. Savitch. Growth functions of stochasticLindenmayer systems. Information and Control, 45:217–228,1980.
[6] T. Yokomori. Stochastic characterizations of EOL languages. Information and Control, 45:26–33, 1980.
[7] G. T. Herman and G. Rozenberg. Developmental systems and languages. North-Holland, Amsterdam, 1975.
[8] A. Lindenmayer and G. Rozenberg, editors. Automata, languages, development. North-Holland, Amsterdam, 1976.
[9] A. Salomaa. Formal languages. Academic Press, New York, 1973.
[10] P. Prusinkiewicz and J. Hanan. Visualization of botanical structures and processes using parametric L-systems. In D. Thalmann, editor, Scientific Visualization and Graphics Simulation, pages 183–201. J. Wiley & Sons, 1990.
[11] A. Aho, R. Sethi, and J. Ullman, *Compilers: principles, reachniques, and tools*, 1986.
[12] J. Termblay and P. Sprenson, *The Theory and Practice of Compiler Writnig*, 1985.
[13] C. Fisher and R. LeBlanc, *Crafting a Compiler with C*, 1991.

زیرنویس‌ها

[1] Turtle interpretation
[2] Chomsky
[3] Bracketed OL-system
[4] Stochastic L-systems
[5] Context-sensitive L-systems
[6] Parametric L-system
[7] Axiom