Реферат на тему:

**Контекстно-вільні та LA(1)-граматики**

**1. Контекстно-вільні граматики**

***Контекстно-вільною***, або ***КВ-граматикою***, називається граматика, в якій ліві частини всіх продукцій є нетерміналами. Зміст терміну "контекстно-вільна" полягає в тім, що застосування продукції *A w* до ланцюжка *uAv* не залежить, тобто є ***вільним*** від сусідніх з *A* символів, які утворюють ***контекст*** *uv*.

Зазначимо, що БНФ вигляду *A*::=*w* цілком аналогічна продукції *A w*. Отже, сукупності БНФ є просто іншою формою КВ-граматик.

***Контекстно-вільною мовою*** (***КВ-мовою***) називається мова, що може бути задана КВ-граматикою.

Прикладами КВ-граматик та КВ-мов є граматики з прикладів **21.3**, **21.5**, **21.6** у попередньому параграфі й задані ними мови. Граматика з прикладу **21.7** не є КВ-граматикою. До речі, мова, задана нею, не є КВ-мовою, оскільки не існує КВ-граматики, яка б її задавала.

КВ-граматики відіграють особливу роль у програмуванні, оскільки ними описується синтаксис практично всіх конструкцій мов програмування. Більше того, він описується КВ-граматиками, продукції яких задовольняють певні структурні обмеження. З використанням цих обмежень було побудовано алгоритми синтаксичного аналізу, час виконання яких прямо пропорційний довжині аналізованого слова. А лінійна складність цих алгоритмів великою мірою зумовила ефективність сучасних систем програмування.

**2. Дві ідеї аналізу**

Заміна нетермінала з лівої частини продукції на її праву називається ***розгортанням нетермінала***, а зворотна заміна – ***згортанням правої частини***. Розглянемо дві стратегії аналізу, основані на згортаннях та на розгортаннях, за допомогою наступного прикладу.

**Приклад 8.** Нехай

*G*0 = ( { *a*, +, \*, (, ) }, { *E*, *T*, *F* },

{ *E*  *E* + *T* | *T*, *T*  *T* \* *F* | *F*, *F*  (*E* ) | *a* },

*E* )

– граматика. Нетермінали *E*, *T*, *F* відповідно є скороченнями слів "*E*xpression", "*T*erm", "*F*actor", тобто "вираз", "доданок", "множник". Вони позначають вирази зі знаками операцій +, \*, доданки та множники в них відповідно.

Виведення слова *a*+*a*\**a* в *G*0 з *розгортанням нетерміналів, перших ліворуч у проміжних ланцюжках*, має вигляд:

*E*  *E*+*T*  *T*+*T*  *F*+*T*  *a*+*T*  *a*+*T*\**F*  *a*+*F*\**F*  *a*+*a*\**F* 

 *a*+*a*\**a*

Тут нетермінали, що розгортаються, підкреслені. Аналіз ланцюжка, що відтворює такі розгортання від початкового символу до термінального слова, називається ***низхідним***, або аналізом "***від верху до низу***".

Тепер розглянемо виведення слова *a*+*a*\**a* з *розгортанням нетерміналів, останніх праворуч*:

*E  E*+*T E*+*T*\**F*  *E*+*T*\**a*  *E*+*F*\**a*  *E*+*a*\**a*  *T*+*a*\**a*  *F*+*a*\**a*

 *a*+*a*\**a*

Проміжні слова в цьому виведенні, записані у зворотному порядку, дістаються згортаннями правих частин продукцій, починаючи з термінального слова. Такі згортання від ланцюжка терміналів до початкового нетермінала граматики відтворюються в процесі ***висхідного аналізу***, або аналізу "***від низу до верху***".

Головною проблемою побудови алгоритмів аналізу в обох випадках є необхідність вибору продукції, застосованої для розгортання чи згортання. Чому, наприклад, у першому виведенні на першому кроці вибирається продукція *E E*+*T*, а не *E T*, а на другому, навпаки, *E T* ? Чому за оберненого виведення в слові *E*+*T*\**F*, в якому є дві праві частини продукцій *E*+*T* і *T*\**F*, саме ланцюжок *T*\**F* згортається в *T*, а не *E*+*T* в *E* ? Тут необхідний вибір зроблено тому, що структура термінального слова була відома заздалегідь. Але, взагалі, структура слова до початку його аналізу невідома, і виникає необхідність перебирати продукції для застосування потрібної.

Теоретично, можна розробити алгоритм аналізу на основі перебирання продукцій, але він буде практично неприйнятним внаслідок його оцінки складності. Один із шляхів до ефективних алгоритмів аналізу полягає в обмеженні структури продукцій і позбавленні від перебирання за рахунок звуження множини КВ-граматик. Далі розглядаються саме такі обмежені граматики та побудова алгоритму аналізу для них, складність якого лінійна.

**3. LA(1)-граматики**

LA(1)-граматики дозволяють вибирати необхідну для розгортання продукцію при низхідному аналізі за першим символом ще не розпізнаної частини слова. "LA(1)" позначає речення "Look Ahead 1 symbol", тобто "подивитися спереду на 1 символ".

Нехай *G*=(*X*, *N*, *P*, *S*) – КВ-граматика, і за словом *w* треба визначити, чи належить воно до *L*(*G*). Нехай *S v*1|…|*vp* – усі продукції з нетерміналом *S* ліворуч. Потрібну для розгортання *S* продукцію *S vi* можна визначити безпосередньо за першим символом слова *w*, якщо *множини перших символів ланцюжків, вивідних із v1, v2, … , vp, не перетинаються*. Взагалі, нехай *am*…*an* – нерозпізнана частина слова, початок якої має виводитися з нетермінала *A*, і *A w*1|…|*wk* – усі продукції з *A* ліворуч. Тоді потрібна для розгортання *A* продукція *A wi* визначається за першим символом *am*, якщо *множини перших символів ланцюжків, вивідних з w1, w2, … , wk, не перетинаються*.

Отже, для кожного нетермінала *A* та кожної правої частини *wi* продукцій *A*  *w*1 | *w*2 | … | *wk* означимо множини

first ( *wi* ) = { *a* | *a*  *X*\* і *wi*  *az* для деякого слова *z* },

first ( *A* ) = first ( *wi* ).

Граматика може мати так звані  ***-продукції*** вигляду *A * . У такому разі, якщо *Av* \**b*1…*bn*, то *b*1 може починати слово, вивідне як з *A*, так і з *v*. Визначити за символом *b*1 , з чого саме виводиться слово, можна за умови, що first(*A*) не містить перших символів слів, вивідних із *v*. Множину цих перших символів ланцюжків, що виводяться з усіх можливих правих контекстів нетермінала *A,* позначимо follo*w* (*A*):

follow ( *A* ) = { *a* | *S*  \* *uAv*, *a*  first ( *v* ) }.

Граматика називається ***LA(1)-граматикою***, якщо:

(1) для кожного нетермінала *A* з продукціями *A w*1|…|*wk* , де *wi * для всіх *i*=1,…,*k*, справджується умова

first ( *wi* )  first ( *wj* ) =  при *i*, *j* = 1, … , *k*, *i*  *j*;

(2) для кожного нетермінала *A* такого, що в *P* є продукція *A*   , справджується

first ( *A* )  follow ( *A* ) =  .

Умови (1), (2) називаються ***LA(1)-умовами***.

Не кожна КВ-мова породжується LA(1)-граматикою, але синтаксис практично всіх конструкцій сучасних мов програмування можна задати LA(1)-граматиками. Досить часто мова "природно" задається не LA(1)-граматикою, але таку граматику для неї можна побудувати.

**Приклад 9.** За продукціями *E E*+*T*|*T*, *T T*\**F*|*F*, *F* (*E*)|*a* граматики *G*0 маємо

first( (*E*) ) = { ( }, first ( *a* ) = { *a* }, звідки

first ( *F* ) = { *a*, ( }; first ( *T*\**F* ) = first ( *F* ), звідки

first ( *T*\**F* )  first ( *F* )   ,

тобто *G*0 не є LA(1)-граматикою. Проте мова виразів L(*G*0) задається еквівалентною LA(1)-граматикою. Побудуємо її.

Із *T* виводяться слова *F*, *F*\**F*, *F*\**F*\**F*, … . Додамо нетермінал *B* та правила *B * |\**FB*, *T FB* замість правил *T F*|*T*\**F*. Маємо

first ( *T* ) = first ( *F* ) = { *a*, ( }, first ( \**FB* ) = { \* },

first ( *B* ) = { \* }, follow ( *B* ) = follow ( *T* ) =

= follow ( *E* ) = { +, ) }.

Отже, продукції *T FB* і *B * |\**FB* не порушують LA(1)-умови. Аналогічно, додавши нетермінал *A*, а замість правил *E E*+*T*|*T* правила *E TA*, *A * |+*EA*, одержимо LA(1)-граматику.

**4. Ліворекурсивні та розширені продукції**

Правило вигляду *A Av* називається ***ліворекурсивним***. Якщо в граматиці є продукції *A Av*|*u*, де *u * , то first(*Av*)=first(*u*)  , і граматика не є LA(1)-граматикою. Але за допомогою цих правил виводяться слова з множини {*u*, *uv*, *uvv*, …}, яка задається регулярним виразом *uv*\* або *u*{*v*}. Замість продукцій *A Av*|*u* запишемо *A u*{*v*}. Продукції з регулярними виразами в правій частині називаються ***розширеними***, як і граматики з такими продукціями. Неважко переконатися, що розширені правила не збагачують множину мов, породжених КВ-граматиками.

**Приклад 10.** Розширена граматика *G*01 із правилами *E T*{+*T*}, *T F*{\**F*}, *F* (*E*)|*a* еквівалентна граматиці *G*0. Фактично РБНФ є іншим записом розширених продукцій, а сукупності РБНФ – іншою формою розширених КВ-граматик.