Analiza składniowa (ang. *parsing*) – proces analizy tekstu, w celu ustalenia jego struktury gramatycznej i zgodności z gramatyką języka

-Analiza zstępująca (top-down)
Może być postrzegana jako próba znalezienia lewego wyprowadzenia (ang. leftmost derivation) strumienia wejściowego, przez szukanie drzew wyprowadzenia, wykorzystując zstępujące rozwijanie zgodnie z daną gramatyką. Znaki są przeglądane od lewej do prawej strony. Parsery LL oraz rekurencyjnie zstępujące są przykładami parserów top-down, które nie radzą sobie z lewostronnie rekurencyjnymi produkcjami. Chociaż uważa się, że prosta implementacja analizy zstępującej nie radzi sobie z bezpośrednią i pośrednią rekursją lewostronną oraz może wymagać wykładniczego czasu i przestrzeni przy gramatykach niejednoznacznych, bardziej zaawansowane algorytmy używane do analizy zstępującej, stworzone przez Frosta, Hafiza oraz Callaghana rozwiązują problem lewostronej rekursji i niejednoznaczności w czasie wielomianowym. Ten algorytm może generować zarówno lewe, jak i prawe wyprowadzenie.
-Analiza wstępująca (bottom-up)
Parser próbuje przepisywać wejście, poprzez zastępowanie znalezionych prawych stron produkcji gramatyki ich lewymi stronami, aż do uzyskania tylko symbolu startowego. Przykładami parserów wstępujących są parsery LR

Analizator składniowy lub parser to program odpowiedzialny za przeprowadzenie analizy składniowej. Zazwyczaj, jest jedną z części translatora lub interpretera, która sprawdza składnię oraz buduje strukturę danych – często jest to rodzaj drzewa składni, drzewa AST lub innej struktury gramatycznej. Parser może pracować na pojedynczych znakach tekstu, lecz często wygodniej jest pracować na całych słowach (podczas analizy zdania). Do wstępnego podzielenie tekstu na słowa (ang. token) używany jest osobny moduł nazywany lekserem lub analizatorem leksykalnym. Parsery mogą być tworzone ręcznie, lecz częściej są generowane przez generatory parserów (np. Yacc) na podstawie opisu gramatyki.

Zastosowanie
Najczęstszym zastosowaniem parserów jest analiza języków programowania. Mają one, zwykle, prostą gramatykę z nielicznymi wyjątkami. Jednakże gramatyki bezkontekstowe mają ograniczone zastosowanie, gdyż mogą one opisać jedynie ograniczony zestaw języków.
Ręczne pisanie parsera, szczególnie dla dużych języków, jest zajęciem dosyć żmudnym, dlatego powstały generatory parserów. Jednym z popularniejszych generatorów jest **yacc**, pozwalający na generowanie parserów w języku C. Jego odpowiednikiem rozprowadzanym na zasadach wolnego oprogramowania jest stworzony przez Free Software Foundation **bison**. Wśród przykładów generatorów

parserów dla innych języków jest ocamlyacc dla języka OCaml oraz **JavaCC** i **SableCC** dla Javy.

LEKSERY

program komputerowy który dokonuje analizy leksykalnej danych wejściowych, zwykle jako pierwsza część jakiegoś większego procesu, np. kompilacji.

Programy generujące leksery to między innymi lex, flex i ocamllex.

Lex/Flex wiadomości podstawowe

Przypomnijmy, że analizą leksykalną nazywamy podział danych wejściowych składających się z ciągu znaków (tekstu, zawartości pliku, itp.) na ciąg określonych symboli leksykalnych nazywanych tokenami. Tokeny opisywane są zazwyczaj za pomocą wyrażeń regularnych. Każde wyrażenie regularne opisuje pewien język – zbiór ciągów znaków dających się do niego dopasować.

Programy wykonujące analizę leksykalną nazywamy lekserami. Program taki może zostać zaimplementowany bezpośrednio w dowolnym języku programowania. Można również wykorzystać w tym celu narzędzia umożliwiające generowanie kodu źródłowego analizatora leksykalnego na podstawie przygotowanej specyfikacji działania. Przykładem tego typu narzędzia jest program lex lub odpowiadający mu w systemie GNU program flex.

Zadanie wstępne: Przypomnij informacje dotyczące analizy leksykalnej oraz programu lex, które poznałeś w trakcie wykładu.

Przykład

Następujący przykład prezentuje typowy przypadek przeprowadzania analizy składniowej prostego języka wyrażeń arytmetycznych, z dwoma poziomami gramatyki: leksykalnym i składniowym.

Faza pierwsza jest generacją tokenów (analizą leksykalną), w trakcie której strumień wejściowy jest dzielony na znaczące symbole, określone przez gramatykę wyrażeń regularnych. Na przykład program kalkulatora mógłby przetwarzać ciąg znaków taki jak „12*(3+4)^2” i dzielić to na symbole 12, *, (, 3, +, 4,), ^, 2, gdzie każdy coś oznacza w kontekście wyrażenia arytmetycznego. Lekser mógłby zawierać reguły, które mówiłyby, że znaki *, +, ^, (,) zaznaczają początek nowego znaku, tak że ciągi bez znaczenia jak „12*” albo „(3” nie byłyby brane pod uwagę.

Następna faza to analiza składni (analiza syntaktyczna), która sprawdza czy symbole tworzą dopuszczalne wyrażenie. To zazwyczaj jest robione w oparciu o gramatykę bezkontekstową, która definiuje rekurencyjnie komponenty tworzące wyrażenie oraz ich kolejność. Jednakże nie wszystkie zasady, definiujące języki programowania mogą zostać wyrażone przy pomocy czystej gramatyki bezkontekstowej, na przykład zgodność typów oraz właściwa deklaracja

identyfikatorów. Te zasady mogą zostać formalnie wyrażone za pomocą gramatyki atrybutywnej.

Faza końcowa to analiza semantyczna, która opracowuje znaczenie dopiero co zweryfikowanego wyrażenia i podejmuje odpowiednie działania. W przypadku kalkulatora, zadaniem jest obliczenie wyrażenia. Kompilator mógłby w tym miejscu generować kod wynikowy. Do definiowana tych akcji może być również użyta gramatyka atrybutywna.