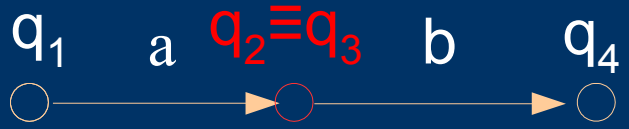
Lexikálna analýza

- Spôsoby popisu lexiky
  - Kleeneho regulárne výrazy
  - Pravo (ľavo) lineárne gramatiky
  - Kombinácia zoznamu rezervovaných slov a niektorého z vyššie uvedených spôsobov
- Implementácia lexikálneho analyzátoru
  - Thomsonova metóda
  - Priama syntéza deterministického konečného automatu
  - Naprogramovanie v programovacom jazyku (napr. C)
  - Neimplementovať — pridať k syntaktickej analýze
- Súvisiace problémy
  - Vyhľadanie vzorky v texte

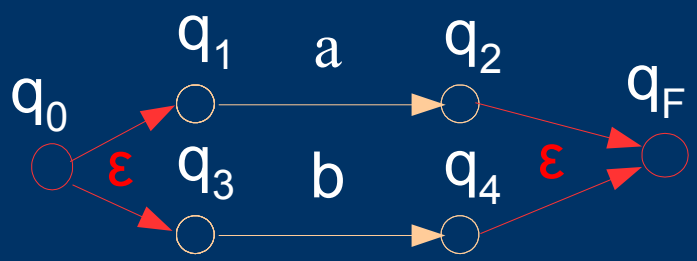
Thomsonova metóda

Kleeneho regulárne výrazy → nedeterministický automat

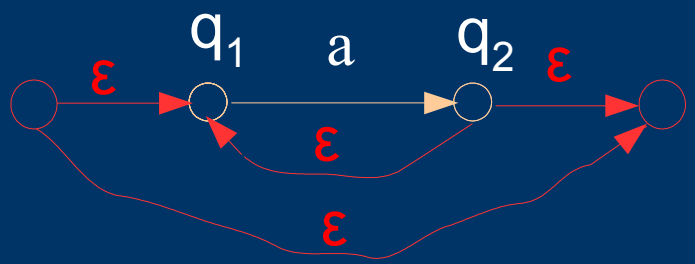
: a



: ab



: a|b



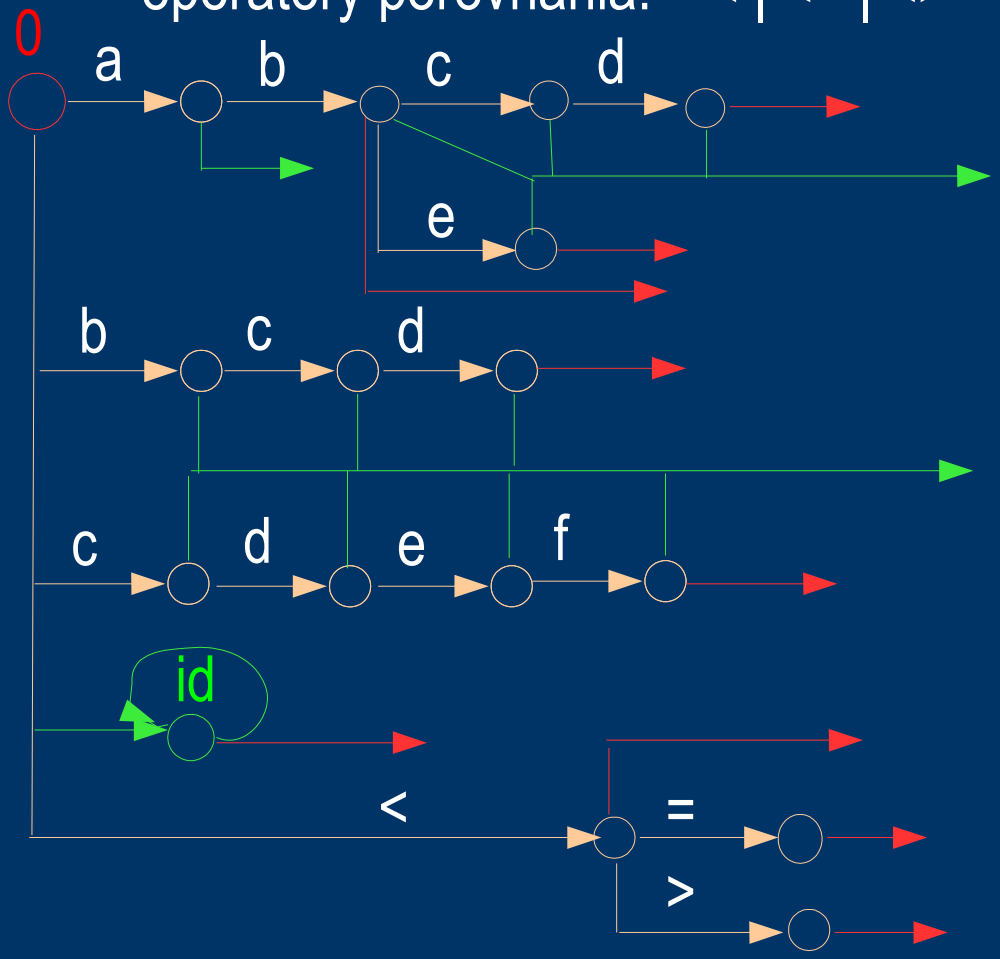
: a*

Písmená a, b podľa potreby reprezentujú jednotlivé znaky reťazce, celé podjazyky resp. automaty ich rozpoznávajúce.

Na vytvorenie deterministického automatu môžeme použiť transformáciu nedeterministického automatu na deterministický, známu z teórie automatov a jazykov. Pretože nedeterministické prechody sú iba na ε, stačí nové deterministické stavy počítať ako ε-uzáver pôvodných stavov.

Priamy návrh deterministického automatu

Príklad: rezervované slová: abcd | abe | ab | bcd | cdef
identifikátory: písmeno{písmeno|číslica}*
operátory porovnania: < | <= | <>



Oranžové šípky prechod do nasledujúceho stavu na znak, ktorým sú označené.
Zelené šípky prechod na písmeno alebo číslicu do stavu id (Zo stavu 0 iba na písmeno). Použije sa iba, ak sa oranžová šípka nedá použiť.
Červená šípka: akceptuje rozpoznaný reťazec a na prázdny symbol prechádza do počiatočného stavu. Použije sa iba, ak sa žiadná iná šípka nedá použiť.

Implementácia deterministického rozpoznávacieho automatu.

Implementácia deterministického rozpoznávača je priamočiara. V jednej premennej si udržujeme stav. „Šípky“ implementujeme pomocou prechodovej tabuľky. Veľkosť tabuľky je ***počet stavov × počet znakov.*** Zaplnenosť tabuľky je relatívne riedká. Navyše často sú prechody na všetky „podobné“ znaky (znaky tej istej kategórie) rovnaké. Pohľad dopredu (lookahead) využívame len v niektorých stavoch, pri ε-prechodoch. Z hľadiska teórie automatov sú to terminálne stavy. Ku každému prechodu môže byť priradená sémantická akcia (program, skupina programov), ktorá sa pred prechodom vykoná.

Vstup ako objekt

- Premenné
  - Prezeraný znak (current symbol)
  - Nasledujúci znak (lookahead)
  - Číslo riadku (line number)
  - Kategória (typ) prezeraného znaku (napr. písmeno, číslica, ...)
- Metódy
  - **Inicializácia:** current symbol := prvý znak súboru;
    lookahead := druhý znak súboru;
  - **Čítanie:** **if** current symbol = \nl **then** line := line + 1;
    current symbol := lookahead;
    lookahead := nasledujúci znak súboru;
    category1 := type(current symbol);
    category2 := type(lookahead);

Kategórie – typy znakov

- Písmená: A|B| ... |Z|a| ... |z|_|@|$
- Číslice: 0|1|2|3|4|5|6|7|8|9
- Biely priestor (blanks): ␣|Tab|LF|CR
- Operátory*
  - Aritmetické: +|-|×|/|div|mod|↑
  - Booleovské: ¬|∧|∨|⊕
  - Porovnávacie: <|>|<=|>=|<>|=
  - Priradenie: :=
- Zátvorky: (|)|[|]|{|}
- Oddelovače: ,|.|;|:

* Presne by sme mali uvádzať len znaky sa v nich vyskytujúce.

Spolupráca s tabuľkou symbolov

Pri skutočnej lexikálnej analýze potrebujeme nielen rozpoznať identifikátor, konštantu, či operátor, ale aj vedieť, ktorý identifikátor či operátor to bol prípadne aj hodnotu rozpoznanej konštanty.
Tieto hodnoty sa ukladajú do tabuľky symbolov.
Počas rozpoznávania identifikátoru musíme v jednej reťazcovej premennej vyskladať jeho hodnotu, výhodné je súčasne v jednej číselnej premennej vypočítame jeho hash.
Pre číselnú konštantu musíme naviac vypočítať jej hodnotu.
Všetko by sme mohli urobiť na základe uloženého reťazca, ale je to zbytočné zdvojnásobenie práce.

Zjednodušenie – rezevované slová ako identifikátor

Prečo komplikovať lexikálnu analýzu množstvom rezervovaných slov? Nie je jednoduchšie predplniť tabuľku symbolov rezervovanými slovami? Ak to urobíme zostane rozpoznávať len štyri základné konštrukcie.

- Identifikátory – rezervované slová
- Konštanty
- Biely priestor
- Zvyšok: operátory, oddelovače, zátvorky – prevažne jedno a dvojznakové konštrukcie

Problémy a komplikácie

- Konštanty so znamienkom: lexikálny analyzátor nie je schopný rozoznať, či ide o znamienko patriace ku konštante alebo aritmetický operátor, ktorý patrí k nadradenému aritmetikému výrazu.
- Biely priestor
  - staré jazyky ho úplne ignorovali
  - súčasné jazyky ako jeden oddelovač
  - číslovanie riadkov a listing
- Perverzné konštrukcie v niektorých jazykoch. Fortran
  - DO 500 I = 1, 25 (fortran úplne ignoruje biely priestor)
  - DO500I = 1.25 (pri použití desatinnej čiarky - nejednoznačné)

Späť k programovaniu

Zdá sa, že pri zjedušenom prístupe programovanie vstupu a sémantických programov výrazne prevyšuje zvyšok lexikálneho analyzátoru. Navyše niektoré jazyky (napr. C) také funkcie ako is_digit is_letter podporujú je preto prirodzené naprogramovať lexikálny analyzátor v takomto jazyku.
Ak programujeme syntaktickú analýzu rekurzívnym zostupom je možné implementovať aj pravidlá pre lexiku v syntaktickej analýze.