Thomson-algoritmus alapú reguláris kifejezés kiértékelő motor implementálása Webkit-be

Hodován Renáta
Programtervező matematikus

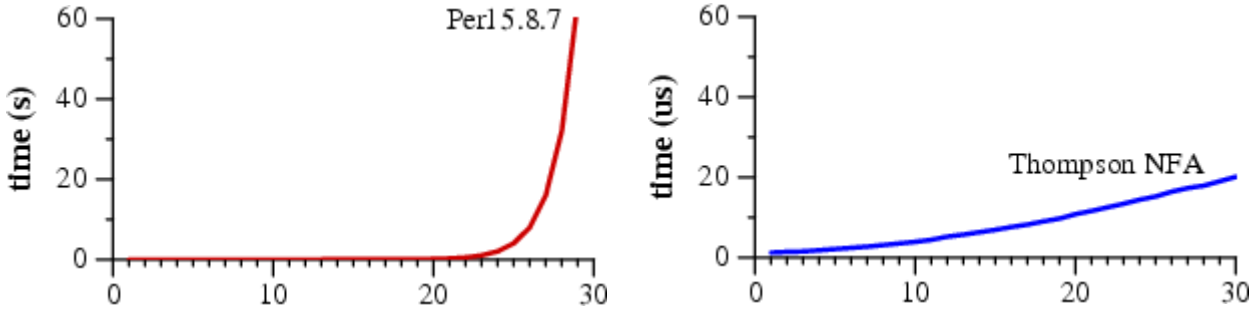
Napjainkban az internet térnyerésének köszönhetően egyre nagyobb a kereslet a felhasználók igényeit minél inkább kielégítő böngészők, böngésző motorok készítésére. Az egyik ilyen böngésző motor a *WebKit*, mely használói között olyan nevek sorakoznak, mint az Apple Safari, Google Chrome, Android, Nokia, stb... A vizuális élmény biztosítása mellett a böngészési sebesség növelése a felhasználók másik legfontosabb elvárása. Mivel a reguláris kifejezések kiértékelése a gyakori feladatok közé tartozik, ezért ennek hatékony megvalósítása különösen hangsúlyos kérdés.

A *WebKit* kezdetben az *ECMA* szabvány által támogatott *PCRE-t* (Perl Compatible Regular Expressions) használta. Ez egy *PERL* szintaktikán és szemantikán alapuló C függvénykönyvtár. Később áttértek a *YARR (Yet Another Regex Runtime)* névre keresztelt, saját fejlesztésű kiértékelő engine-re és a *PCRE-t* csak fallback-ként használják. Mindkét változat egy nemdeterminisztikus véges automatára épül (röviden: *NFA*) és rekurzív algoritmust használ. Ez szemléletesen annyit jelent, hogy az algoritmus olyan automatát épít, ahol adott állapotból ugyanazon betű elolvasásának hatására több különböző állapotba is kerülhetünk. Illesztéskor pedig az összes lehetőséget végigpróbáljuk. Először rápróbáljuk az inputunkat a legelső elágazásra. Ha ez nem vezet eredményre, akkor backtrack-elünk a legutoljára érintett elágazásra és ott a következő, még nem vizsgált alternatívával próbálkozunk. Kellően bonyolult reguláris kifejezés esetén a futásidő exponenciálisra nőhet. Ezt elkerülendő, Ken Thompson a '60-as években kifejlesztett egy algoritmust, melynek nincs szüksége visszalépésekre . Jelen dolgozat ezen megközelítést fogja beépíteni a népszerű böngésző motorba.

A *Thompson-algoritmus* alapvetően abban tér el a fent említettektől, hogy *NFA* helyett *DFA* (Determinisztikus Véges Automata)-t használ. Vagyis adott állapotból egy betű hatására legfeljebb egy másik állapotba mehetünk át és ennek segítségével végezzük el az illesztést.

Gyakorlatban ez úgy fog kinézni, hogy először a Parser által visszaadott *NFA*-nkból készítünk egy alkalmas *DFA*-reprezentációt. A *DFA* segítségével meg tudjuk mondani, hogy mely állapotok érhetőek el egy adott állapotból. Így elegendő csak az illesztés aktuális állapotát tárolni. Tehát az összes lehetséges aktuális állapotot egyszerre tároljuk és megvizsgáljuk, hogy onnan mely mások érhetőek el. Ha elérhető valamelyik végállapot, akkor az illesztés sikeres. Ha nem, akkor az utóbb megtalált elérhető pontokat tekintve aktuális állapothalmaznak addig ismételjük az iménti lépést, míg vagy el nem értük az inputunk végét, vagy meg nem találtuk a szükséges illesztést. Jól látható, hogy nincs szükségünk backtrack-re.

Példaképp tekintsük a^n illesztését $a?^n a^n$ -re. A következő ábrák a PCRE és a Thompson illesztéseinek időtartamát szemléltetik.



A különbség szembetűnő. (Vegyük észre, hogy míg a *PCRE* diagrammja másodperc, addig a Thompson-é mikroszekundum alapú.) Míg a *PCRE*-nek több mint 60 másodpercre van szüksége

egy 29 karakterből álló sztring illesztéséhez, addig a Thomposon-nak mindössze 20 microszekundumra.

Ez már önmagában is nagy lépés a *PCRE*-hez képest. Azonban a *WebKit* rendelkezik *JIT*-támogatással, amit itt is fel tudunk használni. A *JIT Compiler* egy assembly-re emlékeztető szintaxist használó almodul, aminek segítségével futásidőben tudunk gépi kódot előállítani. Lényege, hogy nem kell a köztes reprezentációt minden futáskor értelmezni, mivel az gépi formában van ábrázolva. Ez ismét előny a *PCRE*-vel szemben, habár már a *YARR* is rendelkezik az interpreter mellett *JIT*-támogatással.

Meg kell azonban hagyni, hogy a *Thompsonnal* sajnos nem váltható ki teljes egészében a rekurzív algoritmus. Mivel csak a jelenleg aktív állapotokat tároljuk, nem tudjuk hogyan jutottunk el ide. Pont ez az oka annak, hogy az egyes alminták pozíciójáról nem tudunk információval szolgálni. Ha erre mégis szükség van, akkor a *Thompson* és a *YARR* együttes használatára van szükség. Azonban jó eséllyel még így is kedvezőbb futásidőt tudunk majd elérni, mintha bármelyik korábbi eszközt használnánk.