Scrabble – Programátorská dokumentace

Ondrej Profant

23. prosince 2011

Obsah

1 Použité nástroje

Program je psán v jazyce C# (verze 4) a využívá knihoven GTK#. Pro vývoj bylo použito vývojové prostředí MonoDevelop (2.6) a distribuoaný systém správy verzí GIT (1.7.5.4).

Pro tvorbu grafiky Inkscape (0.48), externí dokumentace je psaná v LaTeXu (TeXLive 2009-13). Vyvíjeno na Ubuntu 11.04–11.10.

2 Zdrojové kódy – obecně

2.1 Získání

Příkazem:

```
git clone git://github.com/Kedrigern/scrabble.git
```

získáte celý projekt. Můžeme ho rozdělit do tří částí:

1. Scrabble: obsahuje samotný kód v C#. Soubory v této složce uvidíte i z Monodevelop, více viz kap. 3.
2. scripts: Obsahuje pomocné skripty, více viz. podkapitola 2.3.
3. DOC-CS: Obsahuje dokumentaci (včetně LaTeX zdrojových kódů)

2.2 Využití IDE

Soubor s koncovkou `sln` lze otevřít v MonoDevelop, které vám zobrazí celou strukturu zdrojových kódů velmi přehledně.

Struktura rozdělení zdrojových kódů snad mluví sama za sebe. Většina důležitých tříd a funkcí je komentovaná přímo v kódu – velká část této dokumentace také vychází rovnou z automaticky vyexportované inline dokumentace.

Pokud by vám MonoDevelop nevyhovalo, tak můžeme bez problémů použít jinou strukturu (např. MS Visual Studio 2010 používá stejnou).

2.3 Pomocné skripty

V adresáři `scripts` je serie shellových skriptů, které pomáhají při kompilaci (např. různé druhy kompilace), balíčkování, získávání slovníků etc.

Skripty jsou psané pro spuštění ve složce `scripts` (relativní cesty).

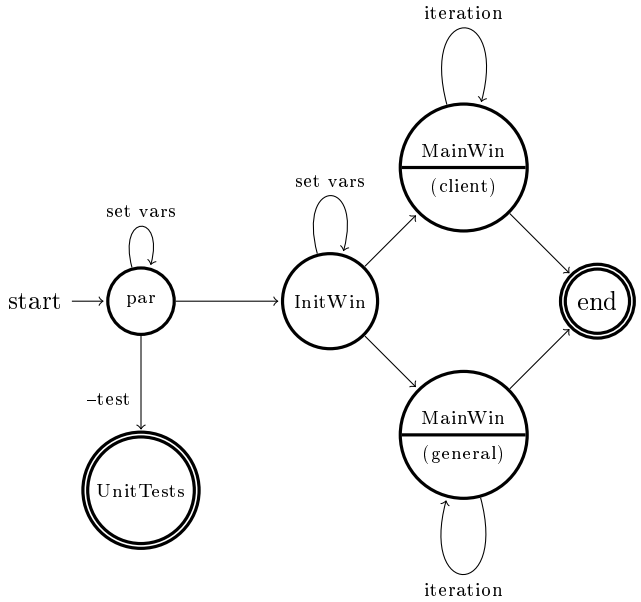
Vybrané:

- `makeDeb.sh` – vytvoří deb balík. Potřebuje práva roota a v `Scrabble/bin/Release` musí být soubor `scrabble.exe`.
- `makeDoc.sh` – vytvoří dokumetaci API v HTMl. Potřebuje abyste zkompilovali projekt (Release) a měli nastavené generováníd okumentace (což zapříčíní vytvoření souboru `scrabble.xml` v `Scrabble/bin/Release`.

3 Zdrojové kódy – Scrabble

3.1 Obecně

Základní běh programu se řídí tímto automatem:



Ve hře jednoho hráče se víceméně vše řídí událostmi hlavního okna. Hráč klikne na pole kam chce zadat slovo. Následně se ověří zda je to přípustné, pokud ano, tak se slovo položí. Pokud tah přípustný není, tak se nic nestane (v plánu je zvuk pro selhání).

Když se je tah korektně položen (hráč odehrál), tak se spustí funkce changeplayer(), která aktivuje dalšího hráče. Pokud je další hráč ovládán člověkem, tak se zobrazí dialog "Na tahu je hráč...". Pokud je dalším hráčem počítač tak rovnou odehraje – vše se odehraje tak rychle, že není nic jiného (blokovat tlačítka) nutné.

3.2 GADDAG - motivace a popis

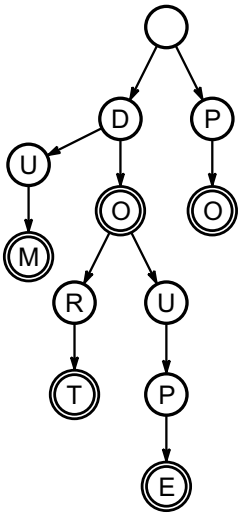
Základní datová struktura a algoritmus je tak klíčová, že jí nejdříve popíšu poněkud obecněji, než je zvykem.

3.2.1 Motivace

Základní klasickou reprezentací slovníku je Trie. V anglickém slovníku o 94 240 slovech má trie 117 150 vrcholů a 179 618 hran.

Existují metody, jak Trii komprimovat, např DAWG (Directed Acyclic Word-Graph), ale vzhledem k dnešním počítačům mi to přijde až zbytečné (a to počítám i s mobilními telefony). Na již zmíněném slovníku se dosáhne velikosti 175 KB oproti 780 KB v normální Trii. Navíc je takováto reprezentace samozřejmě složitější a tím i náchylnější na chyby.

Zajímavější je spíše Trii upravit k našim algoritmickým potřebám. Budeme skládat slova dle modelu:

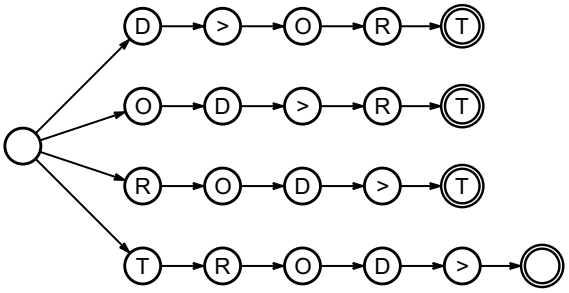


Obrázek 1: Trie pro slovník DŮM, DO, DORT, DOUPĚ, PO

`PREFIX + JIŽ POLOŽENÁ PÍSMENA + SUFIX`

a potřebujeme tedy rychle (snadně) hledat prexfixy k existujícím písmenům, abychom co nejméně omezily zcela nevhodná políčka. Obdobně třeba pracuje Boyer-Moorův vyhledávací algoritmus, který skáče v textu na první pohled trochu nepřehledně, ale může tím ušetři značné prostředky.

3.2.2 GADDAG



Obrázek 2: GADDAG pro slovo DORT

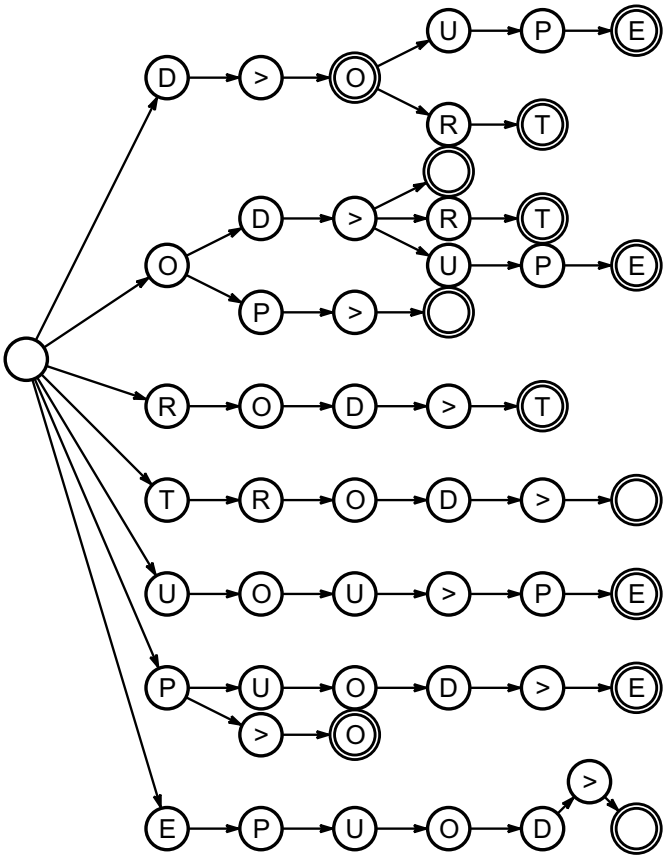
Takovou reprezentaci v roce 1993 představil Steven Gordon, nazval ji GADDAG. Tato struktura je podobná Trii (vychází z ní), aby bylo možné v ní rychle hledat. Slovo se zde dělí na všechny možné prefixy x a zbytek y. Všechny možnosti jsou uloženy. Prefix je uložen zrcadlově, množinově zapsáno („>“ je pouze oddělovač):

$$\{rev(x) > y \mid xy \text{ je slovo jazyka}\}$$

Díky tomu snadno analyzujeme zda se slovo na dané místo hracího plánu hodí. Nejdříve se totiž snažím pokládat prefix zprava doleva (obráceně než jsme zvyklí) a když narazíme na oddělovač, tak doplníme zbytek slova (na „druhém konci“).

GADDAG pro jedno slovo délky n má tedy n cest, jak vidíme na obrázku 2, kde je celý GADDAG pro slovo DORT. Vidíme, že tato struktura je docela rozsáhla. Jaká je její velikost? V případě anglického slovníku je přibližně 5× větší než příslušná trie. Velikost se odvýjí od průměru délky slov ve slovníku, čili v češtině bude velikost obdobná (ne o moc větší).

3.2.3 Algoritmus vyhledávání



Obrázek 3: GADDAG pro slova DŮM, DO, DORT, DOUPĚ, PO

V našem slovníku budeme mít slova: DŮM, DO, DORT, DOUPĚ, PO.[1]

[1] V obrázcích je z technických důvodů vynechána diakritika. Doufám, že to nepovede ke

Slovník reprezentovaný trii je na obrázku 1.

GADDAG pro slovo dort na obrázku 2. Jako oddělovač prefixu jsem zvolil „>". Plný GADDAG pro náš slovník je na obrázku 3.

Pro jednoduchost zatím vynecháme zásobník (rack) a budeme předpokládat, že můžeme umístit libovolný počet libovolných písmen. Myslím, že je vidět, že zásobník následně poskytne jednoduchou heuristiku, která velmi ořeže možnosti.

Z počátku budeme mít na naší hrací ploše slovo DŮM a DO (spojená skrz D). Plochu budeme procházet celou 2×. Prvně pro slova umístěná vodorovně (zleva doprava), podruhé pro slova umístěná zhora dolu.

---

zmatení.

```
   1 2 3 4 5 6        Procházení první:
  +-----------+        A1: Nemá spojení.
A~|. . . . . .|        A2-3: Má spojení na B2, res B3 (popsáno dále)
B |. D O~. . .|        A4-6: Nemá spojení.
C |. Ů . . . .|        B1: Nemá spojení (připojení zleva se řeší z~druhé strany)
D |. M . . . .|        B2-3: Obsazeno (přeskočíme)
  +-----------+        B4: Spojení skrz políčko vlevo - nejzajímavější situace.
```

Při procházení nám zatím nastaly zajímavé situace na políčkách A2, A4 a B4. Na A2 začneme, zkoušíme vložit slovo zprava doleva (tak to umí GADDAG), čili projdeme první úroveň GADDAG a zjistíme, že písmena D, O, R, T, U, P, E mají sice návaznost (lze z nich dále tvořit slovo), ale kontrolou křížení, zjistíme, že ani jedno z těchto písmen nemůžeme položit jako prefix slova DŮM.

U A3 bude situace malinko jiná. Zjistíme, že můžeme položit D nebo P (slovo TO nemáme ve slovníku). Dokonce utvoří celá slova a tak spočítáme jejich hodnoty a uložíme je do možných řešení. Budeme dále postupovat zprava doleva na políčko A2, tentokrát, však budeme hledat slova začínající na D či P. Zde zjistíme, že kdybychom položili D, tak můžeme vpravo pokračovat slovami DOUPE a DORT. DOUPE by se nám nevešlo na hrací plán, ale DORT začínající na A3 je přípustným řešením. U P zjistíme, že na A2 nemůžeme položit U. Nicméně druhá možnost položit P na A3 a O na A4 lze – máme další možné řešení.

Na B4 zjistíme reverzní prefix OD může vpravo pokrčovat na slova DORT a DOUPE (a že sám je slovem). A máme další dvě slova do možných řešení.

Zatím jsme vynechávali zásobník s písmeny, které máme k dispozici. Ten nám rozumně omezí možnosti i u velkých plánů, Také jsme zjednodušili křížení, protože když položím písmeno, tak můžeme v druhém směru také doplnit slovo.

3.3 Podrobná dokumentace tříd

Zde bych si dovolil odkázat na automaticky generovanou dokumentaci.

Ve složce `scripts` najdete `makeDoc.sh`, který vám poskytne nejnovější verzi (vygenerována je do složky: `DOC-CS/htmldoc/`).

Také je přístupná online (ne vždy aktuální):
http://kedrigern.github.com/scrabble/doc

4 Odkazy

git : http://git-scm.com

GitHub : http://github.com

GTK# : http://www.mono-project.com/GtkSharp

Inkscape : http://inkscape.org

Mono : http://www.mono-project.com

MonoDevelop : http://monodevelop.com