Bringing Expert System into Ruby

Jarek Jarcec Čecho <jaroslav.cecho@avg.com>

Důvody vedoucí ke vzniku práce

- Snaha ulehčit práci lidským pracovníkům, automatizací některých úloh
  - Statická analýza jednodušších vzorků malware
- Možnost behaviour detekce na straně klienta
  - Jednodušší možnost detekce malware na základě jeho chování

Expertní systémy

- Počítačový program pro simulaci lidského experta
- Dvě části:
  - báze znalostí (pravidla)
  - rozhodovací algoritmus (jak pravidla aplikovat)
- Rozhoduje se na základě znalostí a vstupů
  - Výstup je rozhodnutí a jeho zdůvodnění
- Příklad:
  - Pravidlo: "Pokud je nějaké zvíře savec, potom i jeho potomek je savec"
  - Vstup: "Pes je savec."
  - Vstup: "Štěně je potomek psa."
  - Výstup: "Štěně je savec."

CLIPS

- Systém pro tvorbu expertních systému
- Vytvořen v NASA v 90. letech
- Nápsán v C
- Dnes jako "public domain" software
- Syntaxe uživatelského rozhraní podobná LISPu:
  - Závorkové "šílenství"
  - (defrule X ?fact <- (savec ?zvire) => (retract ?fact) (assert (zvire ?zvire)))

CLIPS - "reálný" příklad

```
(defrule savci
      (savec ?rodic) ?f <- (potomek ?rodic ?mlade)
      =>
      (assert (savec ?mlade) (retract ?f))
)
(assert (savec pes))
(assert (potomek pes stene))
(run)
(facts)
f-0   (savec pes)
f-1   (savec stene)
```

rbClips

- LISP syntaxe CLIPS se nám nelíbila
- Chtěli jsme CLIPS ovládat z výššího jazyka
  - nejlepé skriptovacího
  - vysoká úrověň abstrakce a "jednoduchost" práce
  - garbage collector
  - možnost použít ostatní knihovny jazyka:
    - přístup do databáze (MySQL)
    - spouštění externích AV skenerů
- Zvažovali jsme skriptovací jazyky
  - perl
  - python
  - ruby

Ruby

- Mladý dynamický skriptovací jazyk
- Autorem je japonec Yukihiro "Matz" Matsumoto
- Plně objektový (na rozdíl od Javy a dalších)
  - zajímavost: I třída sama je objekt (=instance jiné třídy)
- Otevřený (třídy lze libovolně rozšiřovat)
  - podobně jako JavaScript
  - včetně vestavěných tříd
  - včetně jednotlivých instancí tříd
- Čitelná syntaxe
  - žádné "perloviny" jako '$_', '$@', '@_'
  - závorky okolo argumentů method jsou nepovinné na místech kde to není syntakticky nejednoznačné

Ruby II

- Bloky
  - v podstatě anonymní fce
  - nesou si s sebou vazby na proměné (bindings)
- každá jazyková konstrukce je výraz
  - včetně konstrukci if, while, ... (tedy i ony něco vrací)
- Inspirace jazykem scheme
  - metody jsou chápány jako posílání zpravy objektu (ten ji buď rozumí nebo ne)
- Nejen písmena jsou povolena v názvu metod (zpráv)
  - Object#nil?
  - Array#map!

rbClips II

- Dvě možnosti integrace CLIPS do ruby:
  - automaticky nebo ručně
- Automaticky - projekt SWIG
  - pouze zapouzdření volání CLIPS fcí z ruby
  - rychlé
  - "ošklivé", žádné objekty, procedurální chování
- Manuálně:
  - napsat propojující rozhraní ručně
  - velká časová (a finanční) náročnost
  - na výstupu je "krásný" ruby kód (plně objektový)
  - rozhodli jsme se pro tuto možnost (a tuto práci)

Fáze projektu

- Nastudovat API CLIPS pro použití v aplikacích
- Nastudovat API ruby pro možnosti binárních rozšíření
- Navrhnout jak bude výsledná knihovna pracovat
- Naimplementovat
  - Aktivní část
- Otestovat
  - Automatické testy jsou tvořeny z návrhu a testovány už při implementaci

Implemtace detailně

- Hotovo:
  - Template
  - Fact
  - Environment
  - Constraint
  - Rule
- Pracuje se na:
  - reálnějších příkladech a přizpůsobení knihovního API pro jednoddušší použití
- Budnoucnost:
  - Zabudovat podporu pro COOL, objektový jazyk CLIPS

rbClips - ukázka

```
require 'rbclips'
include Clips
rule = Rule.new :savci do |r|
        r.pattern 'savec', :rodic
        r.retract 'potomek', :rodic, :potomek
        r.assert 'savec', :potomek
end
Fact.new('savci', 'pes').save
Fact.new('potomek', %w(pes stene)).save
Base.run
```

Konec

- Dotazy?