Wykorzystanie systemów regułowych do implementacji mechanizmu obsługi zdarzeń.

Kajetan Rzepecki

EIS 2014
7 stycznia 2015

1 Wstęp

Celem projektu jest zbadanie możliwości oraz opłacalności implementacji mechanizmu obsługi zdarzeń w systemie programistycznym z wykorzystaniem systemów regułowych z **wnioskowaniem w przód**.

Mechanizm ów ma za zadanie ułatwić obsługę zdarzeń zachodzących w systemie poprzez umożliwienie definiowania reguł i faktów w sposób deklaratywny i zintegrowany ze składnią i semantyką języka programowania, w którym jest wykorzystywany:

```
(assert!
  (predicate subject object) ;; Dodanie faktu do bazy faktów.
  ...)

(whenever rule               ;; Dodanie reguły reprezentującej zdarzenie
  action                     ;; oraz instrukcji je obsługujących do
  ...)                       ;; bazy faktów.

(declare (foo x y)           ;; Deklaracja funkcji, zawierająca
  (_@ a function)            ;; automatycznie inferowane fakty
  (_@ arity 2)               ;; dotyczące funkcji,
  (@ big-oh 1)               ;; dodatkowe fakty dostarczone przez autora oraz
  (@ equal (foo 2 21) 23))   ;; informacje o kontraktach funkcji.

(define (foo x y)
  (+ x y))
```

2 Analiza problemu

Mechanizm obsługi zdarzeń będzie docelowo wykorzystywany w zastosowaniach Internet of Things - środowisku rozproszonym z wysoką redundancją, gdzie wiele węzłów tworzących klaster udostępnia zbliżone funkcjonalności o nieco różnych charakterystykach.

Na potrzeby projektu, węzłem określana będzie instancja maszyny wirtualnej języka programowania, na której dostępne są **moduły** - zbiory funkcji realizujących jakąś funkcjonalność. Dynamicznie łączące i rozłączające się węzły będą generowały zdarzenia (indukowane przez i składające się z elementarnych operacji modyfikacji bazy faktów) takie jak: połączenie nowego węzła, pojawienie się nowego modułu, czy dowolne zmiany zawarte w kodzie przez porgramistę. Zdarzenia te będą przesyłane do pozostałych połączonych węzłów.

Dzięki zastosowaniu systemu regułowego, moduły wchodzące w skład danego węzła będą mogły reagować na napływające zdarzenia odpowiednio modyfikując swoje zachowanie. W celu obsługi danego zdarzenia definiowana będzie reguła (o dowolnej złożoności), która w momencie spełnienia uruchamiała będzie szereg instrukcji obsługujących zdarzenie.

Wykorzystanie wnioskowania w przód umożliwi definiowanie reguł z wyprzedzeniem - powiązane z nimi instrukcje obsługujące zdarzenie zostaną wykonane dopiero w momencie spełnienia reguły, po dostatecznej modyfikacji bazy faktów.

2.1 Przykład zastosowania proponowanego mechanizmu

Posiadając następujący moduł pobierający dane GPS z czujnika:

```
(define-module gps-default
  (provide gps)

  (declare (get-location)
    (@ tolerance 0.01))

  (define (get-location)
    ;; Code that gets current location.
    ))
```

...oraz następującą applikację z niego korzystającą:

```
(define-module gps-app
  (import 'gps-default)

  (define (use-gps-data)
    (let ((curr-location (gps-default:get-location)))
      ;; Use gps function to do something.
      ))

  (define (update-state)
    ;; Update apps state using latest gps data.
    ))
```

...programista jest w stanie zadeklarować obsługę pojawienia się modułu pobierającego dane GPS z większą dokładnością:

```
(define-module gps-app
  (import 'gps-default)

  (define gps-location-function gps-default:get-location)

  (whenever (and (module-loaded ?node ?module)
                 (declares ?module ?function)
                 (name ?function 'get-location)
                 (tolerance ?function ?tol)
                 (< ?tol 0.01))
    (set! gps-location-function ?node:?module:?function)
    (update-state))

  (define (use-gps-data)
    (let ((curr-location (get-location-function)))
      ;; Use gps function to do something.
      ))

  (define (update-state)
    ;; Update apps state using latest gps data.
    ))
```

…dzięki czemu, po podłączeniu węzła udostępniającego następujący moduł:

```
(define-module gps-vendor
  (provide gps)

  (declare (get-location)
    (@ tolerance 0.0001))

  (define (get-location)
    ;; Code that gets current location.
    ))
```

…system działający na dotychczasowym węźle automatycznie będzie wyświetlał dane z większą dokładnością.

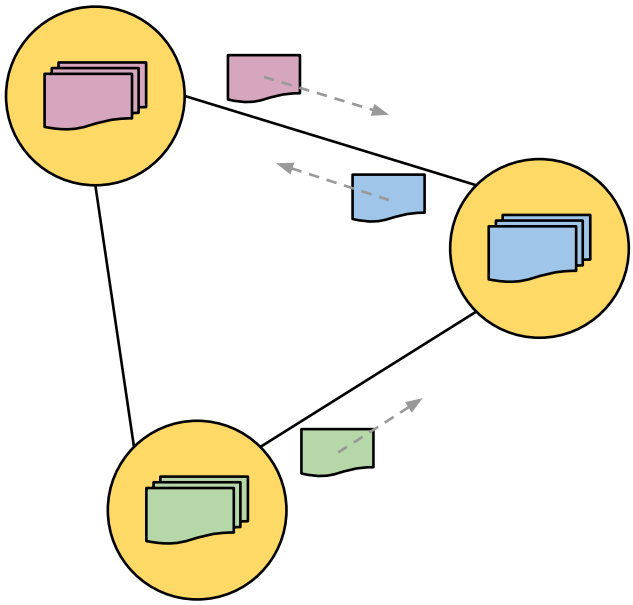
2.2 Analiza możliwości implementacji i przydatności

Przydatność proponowanego mechanizmu jest potencjalnie nieoceniona w domenie języków programowania ponieważ umożliwia ekspresję złożonego przepływu sterowania w deklaratywny sposób - za pomocą krótkich, dobrze zdefiniowanych reguł. Dzięki temu programista tworzący aplikacje wykorzystując system regułowy może skoncentrować się na rezultatach rozwiązania problemu, nie zaś na sposobie ich osiągnięcia - system regułowy zrobi to za niego.

Wykorzystanie systemów regułowych do implementacji systemu modułów języka programowania dodatkowo umożliwia automatyczne i skalowalne tworzenie rozproszonych, dynamicznych systmemów charakteryzujących się dużą redundancją - takich jak Internet of Things. Podejście regułowe zapewnia interfejs komunikacji i mechanizm rozwiązywania konfliktów między poszczególnymi modułami/jednostkami aplikacji, co ułatwia ich kompozycję i umożliwia redundancję.

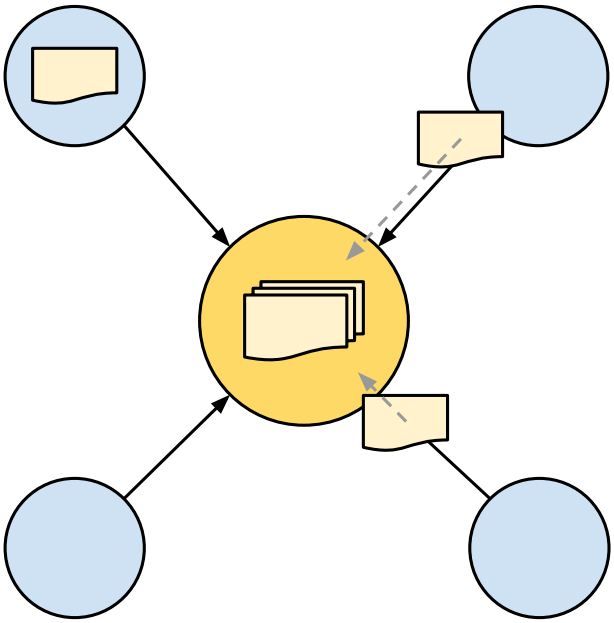
Dodatkowym atutem zastosowania systemów regułowych jest potencjalna skalowalność systemów z nich korzystających - do systemu w każdym momencie można dodać więcej węzłów dostarczających pewne usługi, a system automatycznie będzie w stanie z nich korzystać, reagować na zmiany ich stanu i obsługiwać zdarzenia przez nie sygnalizowane. Zwiększenie ilości węzłów dostarczających te same usługi pozytywnie wpływa także na stabilność i bezpieczeństwo działania systemu dzięki zwiększeniu jego redundancji.

Potencjalne zalety wykorzystania systemów regułowych w systemach rozproszonych szczególnie dobrze widać w idealnym przypadku zaprezentowanym na poniższym diagramie, gdzie każdy węzeł zawiera pewne reguły i generuje zdarzenia przesyłane do pozostałych węzłów systemu:



Taka konfiguracja zapewnia wszystkie opisane powyżej zalety kosztem zasobów wymaganych do implementacji i działania systemów regułowych na każdym węźle.

W przypadku Internet of Things mały rozmiar i oganiczona wydajność pamięciowa/obliczeniowa urządzeń wchodzących w jego skład niestety uniemożliwia stosowanie istniejących, profesjonalnych systemów regułowych w celu implementcji powyższej, idealnej konfiguracji prowadząc do następującego układu:



W tej konfiguracji istnieją dwie klasy węzłów:

- węzły regułowe, których oprogramowanie korzysta z systemów regułowych do obsługi zdarzeń,
- węzły zdarzeniowe, które jedynie generują zdarzenia i przesyłają je do węzłów regułowych systemu.

Ponieważ węzły zdarzeniowe nie umożliwiają definicji reguł, są one zdane na alternatywne, często imperatywne i mało skalowalne sposoby obsługi zdarzeń, istotnym jest więc by proponowany w następujących sekcjach mechanizm obsługi zdarzeń charakteryzował się możliwie niskim narzutem wydajnościowym.

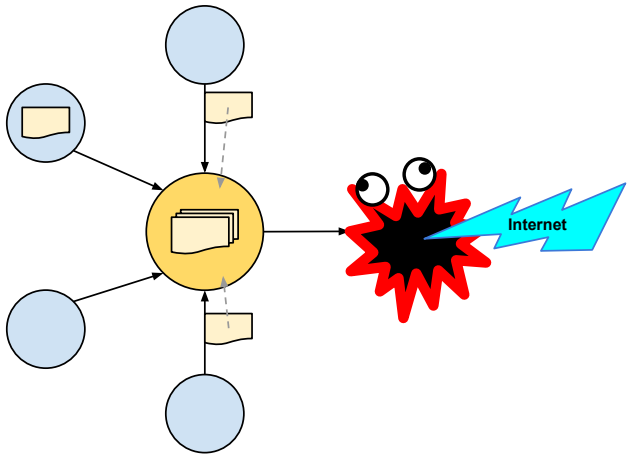
2.3 Podobne rozwiązania

Systemy regułowe wykorzystywane są w wielu różnych dziedzinach, przede wszystkim w systemach ekspertowych do przechowywania i manipulowania wiedzy.

Kluczowym przykładem jest system CLIPS, charakteryzujący się elastycznością - jest on narzędziem do budowy systemów ekspertowych umożliwiającym osadzenie go w gotowej aplikacji. Nie posiada on jednak wszystkich aspektów języka programowania ogólnego przeznaczenia, przez co jego wykorzystanie nie jest dogodne.

Następnym przykładem wykorzystania systemów regułowych do obsługi zdarzeń jest pakiet Drools Fusion, wykorzystujący reguły oraz arbitralny kod w języku Java w kontekście systemów reguł biznesowych (BRMS). Ponieważ jest to system *profesjonalny* i skierowany do dużych korporacji, jest on bardzo rozległy i posiada bardzo wiele, niekoniecznie pożądanych funkcjonalności, przez co jego wykorzystanie w projekcie również nie jest dogodne.

W domenie Internet of Things również powstają rozwiązania oparte o systemy regułowe. Przykładem jest system zaproponowany przez firmę Bosh, który umożliwia zarządzanie inteligentnymi, podłączonymi do IoT urządzeniami z wykorzystaniem reguł. Niestety, jest to system komercyjny i scentralizowany, przez co traci on wiele z zalet zapewnianych przez systemy regułowe:



Klienci mają jedynie możliwość pośredniej ingerencji w tzw. CCU, które są odpowiednikami węzłów regułowych opisanych powyżej. Reguły w CCU modyfikowane są przez zewnętrzny serwis operujący w *Chmurze*, od którego zależy cały budowany system.

3 Szkic rozwiązania

3.1 Porównanie różnych podejść

- Naiwne, iteracyjne algorytmy
- `http://herzberg.ca.sandia.gov/jess/docs/52/rete.html`

3.2 Wybór algorytmów potrzebnych do implementacji

- Rete

4 Prototyp rozwiązania

4.1 Implementacja wybranych algorytmów

4.2 Przykłady zastosowania systemu regułowego

5 Analiza proponowanego rozwiązania

5.1 Analiza wydajności i opłacalności proponowanego rozwiązania

5.2 Wnioski

6 Bibliografia

- Charles L. Forgy, *Rete: A Fast Algorithm for the Many Pattern / Many Object Pattern Match Problem*, Artificial Intelligence 19 (1982), 17-37, `http://dl.acm.org/citation.cfm?id=115736`
- Hesam Samimi, Chris Deaton, Yoshiki Ohshima, Alessandro Warth, and Todd Millstein, *Call by Meaning*, In Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software (Onward! 2014), ACM, New York, NY, USA, 11-28, `http://doi.acm.org/10.1145/2661136.2661152`
- Rule-based Event Management in the Internet of Things and Services