dMARS – specyfikacja architektury agentów

Technologie agenckie

Badania nad technologiami agenckimi mają na celu stworzenie systemów informatycznych, które by w inteligentny sposób wspomagały człowieka w rozwiązywaniu problemów o charakterze rozproszonym lub złożonych obliczeniowo. Przykładami takich problemów są: wyszukiwanie i filtrowanie informacji w sieci WWW, zarządzanie sieciami telekomunikacyjnymi, rynek elektroniczny, wspomaganie zarządzania w przedsiębiorstwie i kontrola ruchu lotniczego.
Badania te, prowadzone od lat osiemdziesiątych, doprowadziły do powstania różnego rodzaju systemów agentów, z których najlepiej w praktyce sprawdził system o nazwie **PRS (Procedural Reasoning System, 1987)**. Architektura ta rozwinęła się później z eksperymentalnej wersji, napisanej w Lispie, do dzisiejszej postaci (w C++), zwanej **"Distributed Multi-Agent Reasoning System" (dMARS)**.

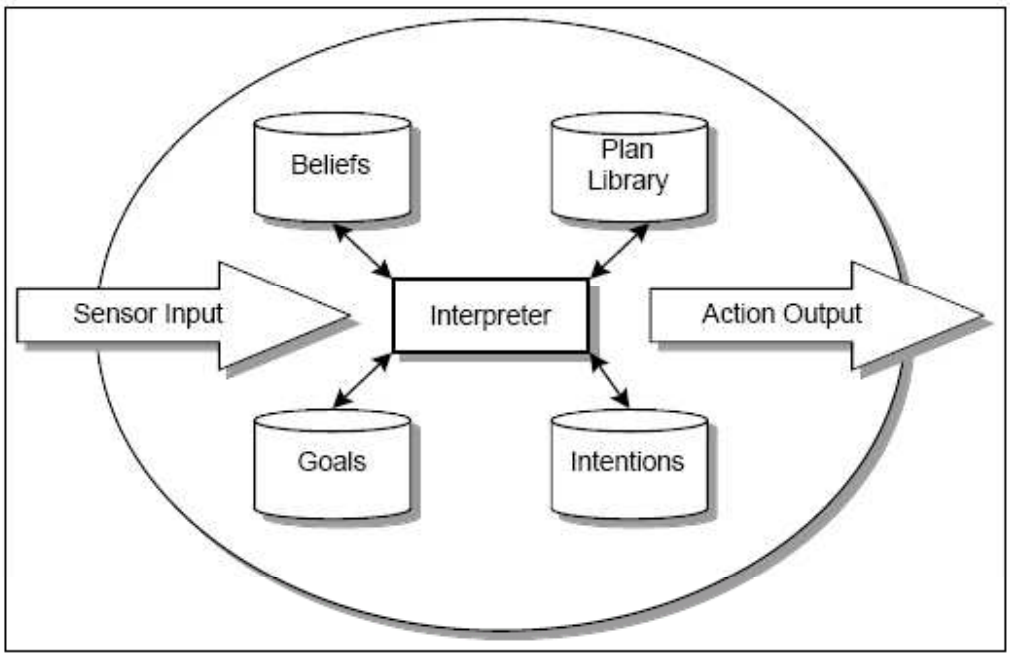
Termin "agent" nie jest dobrze sprecyzowany. Można myśleć o nim w następujący, bardzo ogólny, sposób: jest to system komputerowy osadzony w pewnym środowisku i podejmujący autonomiczne działania mające na celu zrealizowanie jakichś zamierzonych celów. Głównymi postulatami stawianymi przed systemem agenckim są: zdolność działania autonomicznego i zorientowanego na cel, zdolność uczenia się i komunikowania się z innymi agentami. Realizacja tych cech wymaga wyposażenia agenta w struktury przechowywania wiedzy o świecie zewnętrznym i algorytmy jej autonomicznego pozyskiwania, mechanizmy wnioskowania i wykonywania akcji na otoczeniu.

Model BDI (Belief-Desire-Intention)

Podstawą wielu systemów agenckich, w tym również systemu dMARS (a wcześniej - PRS), jest model BDI (*Belief–Desire–Intention*). Jest to model, który nawiązuje w pewnym sensie do sposobu wnioskowania człowieka: opiera się na "przekonaniach", "pragnieniach" i "intencjach". Przekonania wyrażają posiadaną przez agenta wiedzę o otaczającym go świecie, pragnienia – stany świata, które są celami agenta, intencje – rzeczywiste cele agenta, które wybrał spośród dostępnych opcji i do których zobowiązał się dążyć, angażując swoje zasoby. Działanie agenta BDI polega na ciągłym uaktualnianiu swojej bazy przekonań, rozpoznawaniu dostępnych opcji – pragnień, wyborze spośród nich intencji i wykonywaniu akcji w celu ich realizacji.

dMARS

Agent dMARS składa się z biblioteki planów (*plan library*), funkcji wyboru intencji, funkcji wyboru zdarzenia, funkcji wyboru planu i funkcji wyboru podstawienia. Plany specyfikują ciągi akcji, które agent może wykonywać w celu osiągnięcia swoich intencji. Plany zgrupowane są w bibliotece planów, reprezentującej wiedzę proceduralną. Każdy plan składa się między innymi z warunku wywołania (*trigger)*, kontekstu (*pre-condition*), określającego warunki, w których dany plan może być wykonany oraz treści (*body*), definiującej ciąg akcji – elementarnych lub złożonych.



Agent dMARS monitoruje zarówno stan świata zewnętrznego, jak i swój stan wewnętrzny, i umieszcza wszystkie odbierane zdarzenia w kolejce zdarzeń (*event queue*). Składnikami stanu wewnętrznego agenta są: przekonania, intencje oraz zdarzenia. Interpreter zarządza wszystkimi procesami, które są wykonywane w agencie. Działa on w sposób ciągły według następującego cyklu:

- Obserwuje stan świata oraz stan wewnętrzny agenta i aktualizuje kolejkę zdarzeń.
- Generuje nowe, możliwe zadania wyszukując te plany, których warunki wywołania są zgodne ze zdarzeniami w kolejce.
- Wybiera ze zbioru możliwych planów plan do wykonania (ang. intended means).
- Umieszcza wybrany plan na stosie intencji (ang. intention stack).
- Wybiera stos intencji, pobiera plan z jego szczytu i wykonuje kolejny jego krok.

Przekonania, cele, akcje i plany

Przekonania w systemie dMARS przypominają literały z Prologu - są reprezentowane jako zdania logiki pierwszego rzędu (formuły atomowe, z negacją, nie zawierające zmiennych). Zbiór przekonań jest częścią stanu agenta i może się zmieniać w czasie.

Cele są definiowane są także jako formuły, przy użyciu dwóch dodatkowych operatorów "!" oraz "?". Oznaczają one odpowiednio „osiągnięcie” i „sprawdzenie”. Tak więc dla danej formuły X mamy: !X oznacza chęć osiągnięcia X (czyli agent o takim celu rozpocznie ciąg akcji, mających ostatecznie doprowadzić do tego, że X będzie prawdziwe). Natomiast ?X oznacza chęć sprawdzenia, czy X jest prawdziwe. X jest „formułą sytuacyjną” (*situation formula*), tzn. jej prawdziwość zależy od aktualnego zbioru przekonań agenta.

Akcje: akcje mogą być "zewnętrzne" (dotyczące środowiska, w którym działa agent) i „wewnętrzne” (dotyczące samego agenta). Akcje wewnętrzne operują na zbiorze przekonań agenta - może to być albo dodanie nowego "przekonania", albo usunięcie go. O akcjach "zewnętrznych" można myśleć jak o wywołaniach funkcji, które wpływają na środowisko działania agenta.

Plany: plan składa się z następujących części:

- Warunek wywołania i kontekst: definiują warunki, które muszą być spełnione, aby plan mógł zostać uruchomiony. Warunkiem wywołania może być:
    - Nabycie nowego „przekonania”
    - Usunięcie przekonania
    - Przyjęcie nowego celu

  Kontekst planu definiuje zbiór przekonań, które musi mieć agent, aby plan został uruchomiony.
- Treść planu *(plan body)*: to drzewo reprezentujące akcje, które należy wykonać w ramach planu
- „Warunek wykonywania planu” *(maintanance condition):* musi być spełniony przez cały czas wykonywania planu
- Zbiór wewnętrznych akcji, które zostaną wykonane w razie sukcesu planu
- Zbiór wewnętrznych akcji, które zostaną wykonane, gdy plan się nie powiedzie

„Treść planu” jest drzewem, w którym krawędzie (*branches*) reprezentują albo cel *(subgoal)*, albo akcję (wewnętrzną lub zewnętrzną). Pomyślne wykonanie planu następuje, gdy w drzewie-treści planu zostanie osiągnięty którykolwiek z liści.

Uruchamianie planu

Po odczytaniu zdarzenia z kolejki zdarzeń, system wyszukuje potencjalne plany do wykonania. W tym celu, dla każdego planu z biblioteki planów, sprawdza, czy są spełnione warunki wywołania. W ten sposób powstaje lista potencjalnych planów do wywołania. Następnie spośród nich wybierany jest jeden, który zostanie rzeczywiście uruchomiony. Zostaje powołana instancja tego planu (która zawiera dodatkowe informacje, np. aktualną pozycję w drzewie planu – początkowo jest to korzeń drzewa, w miarę realizacji planu pozycja ta będzie się zmieniać).
Instancja planu zostaje umieszczona na szczycie stosu wykonywanych planów. Może być wiele stosów – częścią interpretera jest funkcja wybierająca w każdym kroku, który stos będzie używany. Każdy ze stosów nazywany jest „intencją”.

„Stan” agenta

Możemy teraz sprecyzować, czym jest „stan” agenta. Na stan agenta składają się następujące rzeczy:

- Zbiór przekonań
- Intencje (stosy planów)
- Zdarzenia (kolejka zdarzeń)

Są to te elementy, które będą się zmieniać w czasie. Dodatkowo agent ma jeszcze kilka właściwości, które są niezmienne:

- Bibliotekę planów
- Funkcje wyboru intencji w każdym kroku
- Funkcję wyboru planu
- Funkcję wyboru zdarzenia do przetwarzania
- Funkcje do wyboru ścieżki w drzewie planu oraz funkcję wybierającą właściwe podstawienia (unifikację termów) w czasie realizacji planu

Przetwarzanie zdarzeń

W każdym kroku działania agenta mamy dwie możliwe sytuacje: albo kolejka zdarzeń jest pusta, albo są jakieś zdarzenia czekające na przetworzenie. W tym drugim przypadku wybierane jest zdarzenie, dla niego tworzony plan i dodawany do któregoś ze stosów (intencji).
Jeżeli kolejka zdarzeń jest pusta, przechodzimy do wykonywania intencji. Wybierana jest intencja do wykonania, z niej – wybierany plan (zazwyczaj będzie to plan z wierzchołka stosu), a w ramach planu – krawędź. Do tego służą opisane powyżej funkcje wyboru.

Podsumowanie

W architekturze dMARS agent w ogóle nie wykonuje samodzielnego planowania, lecz wybiera gotowe plany, które muszą być utworzone na etapie projektowania i umieszczone w bibliotece planów. Przekonania agenta wyrażane są podobnie, jak w języku Prolog, tzn. w formie literałów logiki pierwszego rzędu. Cele agenta mogą być dwóch rodzajów: cele do osiągnięcia (*achieve*) i cele do sprawdzenia (*query*). Architektura dMARS posiada formalną specyfikację w języku „Z”, która precyzyjnie definiuje pojęcia intencji, celu, planu itd., jak również przepływ sterowania w interpreterze.