Agenty inteligentne (*Intelligent Agents*)

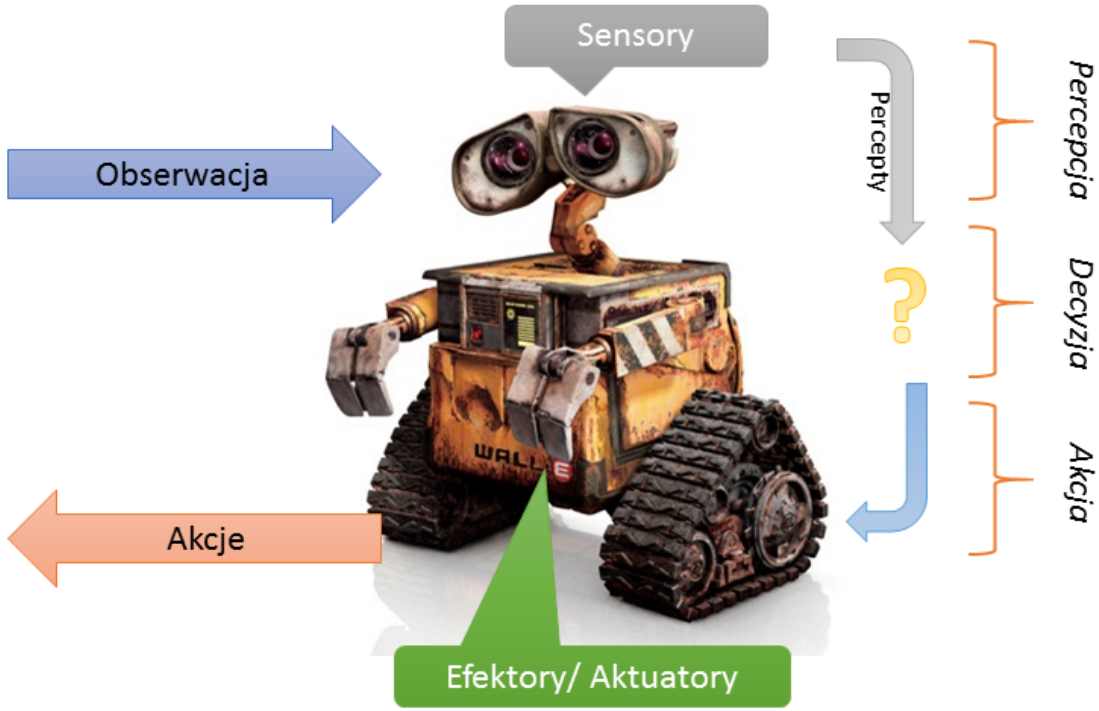
Na podstawie "An Introduction to MultiAgent Systems" oraz slajdów Michaela Wooldridge'a

Definicja agenta

Agent jestem **systemem komputerowym** zdolnym do **autonomicznego działania** w pewnym **środowisku**, aby osiągnąć **przekazane** mu **cele**.

Agent prowadzi ciągła interakcję ze środowiskiem, w którym funkcjonuje
obserwuj $\rightarrow$ decyduj $\rightarrow$ działaj $\rightarrow$ obserwuj $\rightarrow$ decyduj $\rightarrow$ ...

Agent i środowisko



Agent i środowisko

- W większości dziedzin agent nie ma *pełnej kontroli* nad środowiskiem (w najlepszym przypadku jest to częściowa kontrola)
- Ta sama akcja wykonana w takich samych (pozornie) warunkach może mieć zupełnie różne efekty, może się także zakończyć *niepowodzeniem*
- Agent musi być przygotowany na ewentualność niepowodzenia wykonanej akcji

Właściwości środowiska (1)

Dostępne $\leftrightarrow$ niedostępne (*accessible* $\leftrightarrow$ *inaccessible*)

- W środowisku dostępnym agent może uzyskać kompletną, dokładną i aktualną informację o stanie środowiska
- Im bardziej dostępne środowisko, tym łatwiej zbudować agenta, który będzie w nim działał
- ...ale już środowiska o umiarkowanej złożoności są niedostępne

Właściwości środowiska (2)

Deterministyczne $\leftrightarrow$ niedeterministyczne
(*deterministic* $\leftrightarrow$ *nondeterministic*)

- W środowisku deterministycznym każda akcja ma jednoznacznie określony efekt (brak niepewności związanej z wykonaniem akcji)
- Środowiska niedeterministyczne (np. świat fizyczny) są większym wyzwaniem dla twórców agentów

Właściwości środowiska (3)

Statyczne $\leftrightarrow$ dynamiczne (*static* $\leftrightarrow$ *dynamic*)

- Środowisko statyczne zmienia się jedynie pod wpływem akcji agenta
- W środowisku dynamicznym funkcjonują inne procesy (także inne agenty), które wpływają na jego stan niezależnie od działań rozważanego agenta
- Świat fizyczny jest środowiskiem (bardzo) dynamicznym

Właściwości środowiska (4)

Dyskretne $\leftrightarrow$ ciągłe (*discrete* $\leftrightarrow$ *continuous*)

- Środowisko jest dyskretne, jeśli liczba możliwych akcji i obserwowanych stanów jest stała i skończona
- Środowisko gry w GO jest dyskretne, natomiast środowisko „drogowe” ($\rightarrow$ prowadzenie samochodu) jest ciągłe

Pojęcie autonomii

- Autonomia to najważniejsza cecha agenta (chociaż nie świadczy o jego inteligencji)
- Autonomia obejmuje całe spektrum
  - ... od pełnej „wolności” w przyjmowaniu celów, przekonań (dotyczących środowiska) oraz akcji
  - ... do realizacji otrzymanych żądań w ściśle określony sposób (agenty usługowe, *service agents*)

W przypadku agentów autonomię rozumiemy jako samodzielny wybór tych akcji, które pozwalają na realizacje przekazanego celu ($\rightarrow$narzucony cel, ale „wolność” w planowaniu akcji)

Regulowana autonomia

Kontrola (podejmowanie decyzji) przekazywana przez agenta człowiekowi w pewnych specyficznych warunkach:

- kiedy agent jest przekonany, że człowiek podejmie „lepszą” decyzję
- kiedy agent nie posiada umiejętności/kompetencji pozwalających na samodzielne podjęcie decyzji
- kiedy jest bardzo duża niepewność związana ze środowiskiem
- kiedy konsekwencje decyzji mogą spowodować szkody

Decyzje i akcje

- Agent ma zestaw akcji ($\rightarrow$zdolność do modyfikowana środowiska)
- Akcje są związane z warunkami wstępnymi (*preconditions*), które opisują warunki stosowalności akcji (możliwe stany środowiska, w których można jest zastosować)
- Dla agenta kluczowym problemem jest podjęcie decyzji, którą akcję wybrać i wykonać, aby jak najlepiej spełnić przekazany cel

Architektury agentów

Architektury (programowe) dla systemów podejmowania decyzji, które są osadzone w pewnym środowisku

Przykłady (prostych) agentów

- Systemy sterowania
  - Każdy system sterowania można uznać za agenta
  - Trywialny system $\rightarrow$ termostat (cel: utrzymanie pewnej temperatury, akcje: wyłączenie lub wyłącznie ogrzewania)
  - Bardziej złożone systemy z rozbudowanymi strategiami/modelami decyzyjnymi
- Procesy (*daemons*) programowe
  - Każdy program, który monitoruje i modyfikuje środowisko programowe można uznać za agenta
  - Proste przykłady: filtry anty-spamowe, programy anty-wirusowe

Inteligentne agenty

Inteligentny agent powinien demonstrować 3 rodzaje zachowania (w celu realizacji/osiągnięcia przekazanych mu celów):

1. **Reaktywność** (*reactivity*) – zdolność do obserwacji środowiska i do szybkiego reagowania na zachodzące zmiany
2. **Proaktywność** (*pro-activeness*) – zdolność do przejęcia inicjatywy w celu realizacji celu (zachowanie sterowane celem, *goal-driven behavior*)
3. **Zdolności społeczne** (*social abilities*) – zdolność do prowadzenia interakcji z innymi agentami (oraz ludźmi)

Reaktywność i proaktywność

Ważne (i jednocześnie trudne) jest osiągnięcie odpowiedniej równowagi pomiędzy reaktywnością i proaktywnością

- Agenty powinny osiągać swoje cele w sposób systematyczny, np. tworząc i wykonując złożone plany
- ... ale nie powinni wykonywać tych planów „na ślepo” bez kontroli środowiska (plan może nie zadziałać, cel może stracić sens)
- W takich sytuacjach agenty powinny odpowiednio zareagować i np. zmodyfikować plan
- ... ale agenty nie powinny skupiać się tylko na działaniu reaktywnym, aby nie stracić możliwości realizacji głównego celu

Zdolności socjalne

- „Prawdziwy” świat to środowisko wieloagentowe – agenty mogą osiągnąć niektóre swoje cele tylko wchodząc w interakcje z innymi agentami
- Zdolności socjalne to zdolności do
  - **kooperacji/współpracy** (*cooperation*) – pracy razem w grupie w celu osiągnięcia wspólnego celu
  - **koordynacji** (*coordination*) – zarządzania zależnościami pomiędzy akcjami wykonywanymi przez różnych agentów
  - **negocjacji** (*negotiation*) – osiągania porozumienia we wspólnych kwestiach
- W najprostszej sytuajci zdolności socjalne to zdolność do komunikacji (wymiany wiadomości)

Agenty i obiekty (1)

Czy agenty i obiekty to nie to samo?

- Obiekty zawierają (*encapsulate*) pewien stan
- Obiekty komunikują się wymieniając wiadomości
- Obiekty oferują metody odpowiadające operacjom, jakie mogą być użyte, aby wpłynąć na ich stan

Agenty i obiekty (2)

- Agenty zakładają znacznie wyższy poziom autonomii – w szczególności same decydują, czy wykonywać akcje na żądanie innych agentów

Objects do it for free; agents do it because they want to (or do it for "money")...

- Agenty są zdolne do elastycznego (reaktywnego, proaktywnego i społecznego) zachowania – standardowe modele obiektowe nie zajmują się tym (operują na niższym „poziomie abstrakcji")
- MAS są z natury wielowątkowe – każdy agent posiada co najmniej jeden wątek

Agenty jako systemy intencjonalne

- Przypisanie agentom stanów mentalnych (przekonań, życzeń, obaw, ...) w celu wytłumaczenia i zrozumienia ich zachowania ($\rightarrow$postawa intencjonalna, *intentional stance*)
- Możliwość budowy wysokopoziomowych (abstrakcyjnych) modeli dla złożonych systemów, gdzie reprezentacja niskopoziomowa staje się niepraktyczna
- Agenty jako systemy intencjonalne (*intentional systems*) stanowią kolejny i bardziej ogólny paradygmaty projektowania i tworzenia systemów

Rosnący poziom abstrakcji: abstrakcja proceduralna $\rightarrow$ abstrakcyjne typy danych $\rightarrow$ obiekty $\rightarrow$ **agenty**

Ogólna architektura agenta

- Zakładamy, że środowisko może być w dowolnym (dyskretnym) stanie ze skończonego zbioru E

$$E = \left\{ e, e^{'}, \ldots \right\}$$

- Agent posiada zestaw akcji Ac, za pomocą których może zmieniać stan środowiska

$$Ac = \left\{ \alpha, \alpha^{'}, \ldots \right\}$$

- Przebieg (*run*) agenta w środowisku to sekwencja przeplatających się stanów środowiska oraz akcji

$$r : e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \xrightarrow{\alpha_2} e_3 \xrightarrow{\alpha_3} \cdots \xrightarrow{\alpha_{u-1}} e_u$$

Możliwe przebiegi

- Niech $\mathscr{R}$ oznacza zbiór wszystkich możliwych skończonych przebiegów dla zbioru stanów E oraz zbioru akcji Ac
- Niech $\mathscr{R}^{Ac}$ oznacza podzbiór wszystkich tych przebiegów, które kończą się akcją
- Niech $\mathscr{R}^{E}$ oznacza podzbiór wszystkich tych przebiegów, które kończą się stanem środowiska

Środowisko (1)

- Zachowanie się środowiska opisane jest za pomocą funkcji transformacji stanu (*state transformer function*)

$$\tau : \mathscr{R}^{Ac} \rightarrow \mathbf{2}^{E}$$

- Środowiska są
  - zależne od historii – nowy stan jest (częściowo) zdeterminowany przez sekwencję wcześniejszych stanów i akcji
  - niedeterministyczne – występuje niepewność co do wyniku danej akcji
- Jeśli $\tau(r) = \varnothing$, to nie występuje kolejny stan dla r – przebieg się *kończy*

Środowisko (2)

Środowisko *Env* jest formalnie zdefiniowane jako trójka

$$Env = \langle E, e_0, \tau \rangle$$

gdzie E jest zbiorem możliwych stanów, $e_0 \in E$ jest stanem początkowym środowiska, a τ jest funkcją transformacji stanu (albo funkcją przejścia)

Agent

- Agent jest reprezentowany przez funkcję mapującą przebieg na akcję

$$Ag : \mathscr{R}^E \rightarrow Ac$$

- Agent w sposób deterministyczny wybiera kolejną akcję do wykonania biorąc pod uwagę zaobserwowane do tej pory zachowanie systemu (dotychczasowy przebieg)
- Niech $\mathscr{AG}$ oznacza zbiór wszystkich możliwych agentów

System

- System jest zdefiniowany jako para agent–środowisko $\langle Ag, Env \rangle$
- Każdy system jest związany ze zbiorem możliwych przebiegów agenta Ag w środowisku Env oznaczonym jako $\mathscr{R}(Ag, Env)$
- Zakładamy, że $\mathscr{R}(Ag, Env)$ zawiera tylko przebiegi, które się skończyły $(\tau(r) = \varnothing)$

Przebieg

Sekwencja

$$(e_0, \alpha_0, e_1, \alpha_1, e_2 \cdots)$$

reprezentuje przebieg agenta Ag w środowisku $Env = \langle E, e_0, \tau \rangle$ jeśli

1. e_0 jest stanem początkowym w Env
2. $\alpha_0 = Ag(e_0)$
3. dla $u > 0$, $e_u \in \tau(e_0, \alpha_0, \cdots, \alpha_{u-1})$ i $\alpha_u = Ag(e_0, \alpha_0, \cdots, e_u)$

Agent w pełni reaktywny

Agent w pełni reaktywny (*purely reactive*) wybiera akcję do wykonania bez uwzględnienia historii – tylko na podstawie aktualnego stanu środowiska

- Formalnie, agent w pełni reaktywny (albo „czysto" reaktywny) jest reprezentowany jako

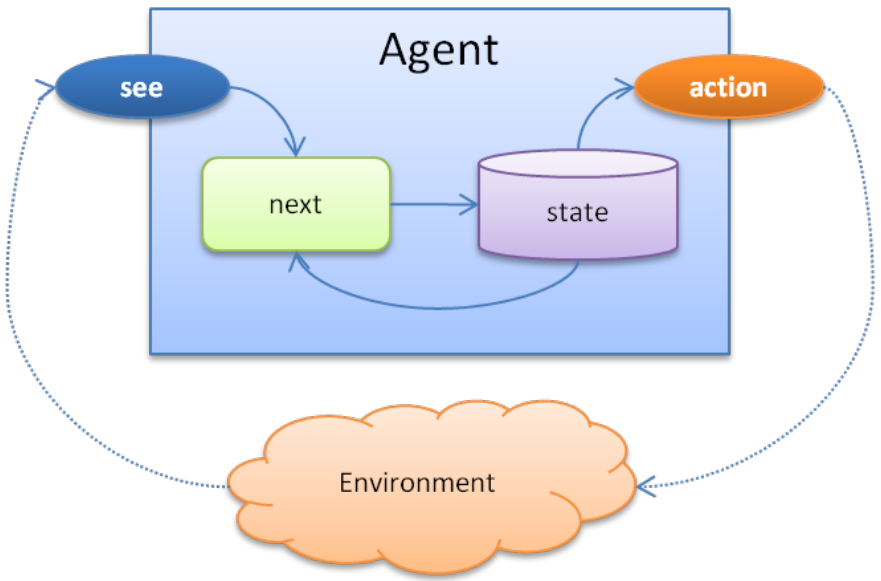
$$Ag : E \rightarrow Ac$$

- Termostat jest przykładem „czysto reaktywnego agenta

$$action(e) = \begin{cases} off & \text{if } e = \text{temperature OK} \\ on & \text{otherwise} \end{cases}$$

Agenty ze stanem (1)

- W celu budowy rzeczywistych agentów uszczegółowienie ogólnego modelu i wprowadzenie komponentów związanych ze sterowaniem i przechowywaniem danych
- Te komponenty tworzą architekturę agenta (i mogą być realizowane na różne sposoby)



Agent ze stanem (2)

- Agent posiada pewną strukturę danych – stan wewnętrzny (*state*) używany do zapisu historii. Niech I będzie zbiorem wszystkich możliwych stanów wewnętrznych agenta
- Funkcja percepcji *see* reprezentuje zdolność agenta do pozyskiwania informacji ze środowiska i transformowania jej do perceptu (obiektu percepcji, *percept*)

$$see : E \rightarrow Per$$

Agent ze stanem (3)

- Funkcja *next* ustala nowy stan agenta na podstawie jego aktualnego stanu oraz perceptu

$$next : I \times Per \rightarrow I$$

- Funkcja wyboru akcji jest zdefiniowana jako

$$action : I \rightarrow Ac$$

Pętla kontrolna agenta

1. Rozpocznij działanie z początkowym stanem wewnętrznym i_0
2. Obserwuj stan środowiska i wygeneruj percept $see(e)$
3. Aktualizuj swój stan wewnętrzny za pomocą funkcji *next* – zmiana na $i_1 = next(i_0, see(e))$
4. Wykorzystaj funkcję *action* do wyboru kolejnej akcji – $action(i_1)$
5. Wykonaj wybraną akcję
6. Wróć do kroku 2

Zadania dla agenta

- Agent jest tworzony, aby wykonywać zadania (realizować cele) dla użytkownika
- Zadania (cele) muszą być podane przez użytkownika...
- ...ale chcemy powiedzieć agentowi, **co** ma zrobić bez tłumaczenia **jak** to zrobić

Użyteczności dla stanów (1)

- Powiązanie użyteczności z poszczególnymi stanami środowiska – zadaniem agenta jest osiągnięcie stanów, które maksymalizują użyteczność
- Specyfikacja zadania w formie (prostej) funkcji użyteczności (*utility function*)

$$u : E \to \mathbb{R}$$

Użyteczności dla stanów (2)

- Ale jak zdefiniować użyteczność dla całego przebiegu?
  - podejście pesymistyczne – użyteczność najgorszego stanu w przebiegu
  - podejście optymistyczne – użyteczność najlepszego stanu w przebiegu
  - suma użyteczności stanów w przebiegu
  - średnia użyteczność stanów w przebiegu
- Wada: ocena lokalna bez możliwości oceny przebiegu w dłuższej perspektywie czasowej

Użyteczności dla przebiegów

- Inne rozwiązanie: przypisanie użyteczności nie do poszczególnych stanów, ale do całych przebiegów

$$u : \mathscr{R} \to \mathbb{R}$$

- To podejście pozwala na uwzględnienie perspektywy długookresowej

Użyteczność w Tileworld (1)

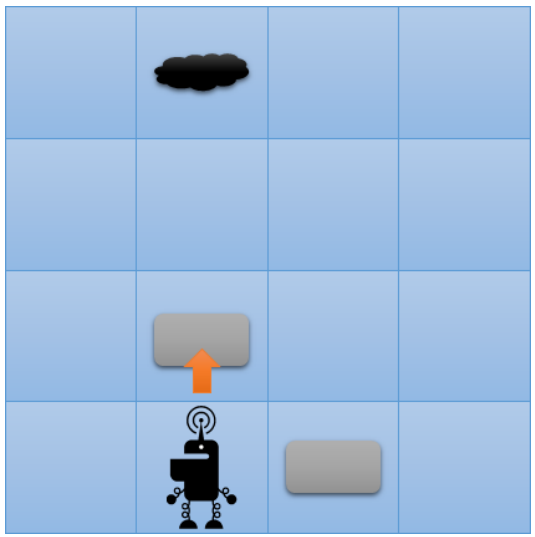
- Symulowane środowisko na dwuwymiarowej planszy z agentami, kafelkami, przeszkodami i dziurami
- Agent może poruszać się w 4 podstawowych kierunkach i jeśli znajduje się obok kafelka, może go popychać
- Dziury muszą być zakrywane kafelkami – agent dostaje za to punkty i jego celem jest zakrycie maksymalnej liczby dziur
- Środowisko zmienia się – dziury pojawiają się i znikają w sposób losowy

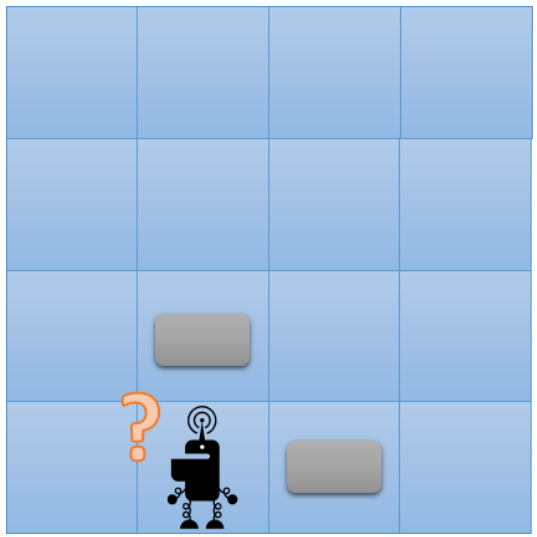
Użyteczność w Tileworld (2)

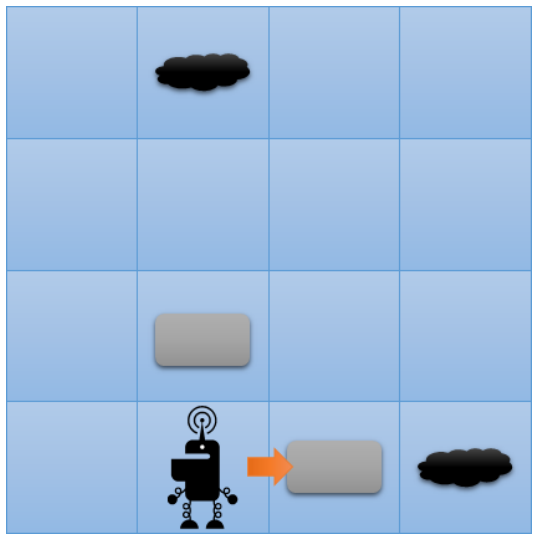
Dany przebieg r jest oceniany za pomocą następującej funkcji użyteczności

$$u(r) = \frac{\text{liczba dziur zakrytych w } r}{\text{łączna liczba dziur w } r}$$

Zmiany środowiska w Tileworld







Oczekiwana użyteczność

- Niech $P(r \mid Ag, Env)$ oznacza prawdopodobieństwo wystąpienia przebiegu r dla agenta Ag w środowisku Env

$$\sum_{r \in \mathscr{R}(Ag, Env)} P(r \mid Ag, Env) = 1$$

- Oczekiwana użyteczność agenta Ag w środowisku Env (dla danego P i u) wynosi

$$EU(Ag, Env) = \sum_{r \in \mathscr{R}(Ag, Env)} u(r) P(r \mid Ag, Env)$$

Agent optymalny

- Agent optymalny Ag_{opt} w środowisku Env to agent, który maksymalizuje oczekiwaną użyteczność

$$Ag_{opt} = \underset{Ag \in \mathscr{AG}}{\arg\max}\, EU(Ag, Env)$$

- Agent optymalny nie musi być zawsze najlepszy – jednak zazwyczaj jest!

Problemy z funkcją użyteczności

1. Dla większości problemów zdefiniowanie funkcji użyteczności jest nietrywialne (lub wręcz niemożliwe)
2. Intuicyjnie wygodniej jest rozważać „cele do osiągnięcia” niż użyteczności

Pomimo tych problemów użyteczności sprawują się dobrze dla pewnych (mniej złożonych) scenariuszy (np. Tileworld)...

Predykatowa specyfikacja zadania

- Specjalny przypadek funkcji użyteczności, która przypisuje przebiegowi wartość 0 lub 1
  - jeśli przebieg uzyskuje ocenę 1, wówczas agent odnosi *sukces*
  - w przeciwnym razie agent ponosi *klęskę*
- Taką funkcję użyteczności określa się jako predykatową specyfikację zadania (*predicate task specification*) i definiuje jako

$$\Psi : \mathscr{R} \rightarrow \{0,1\}$$

Środowisko zadania (1)

- Środowisko zadania (*task environment*) to para $\langle Env, \Psi \rangle$, gdzie Env jest środowiskiem, a Ψ jest predykatową specyfikacją zadania
- Niech $\mathscr{TE}$ będzie zbiorem wszystkich środowisk zadania
- Środowisko zadania definiuje
  - właściwości środowiska, w którym będzie działał agent
  - kryteria oceniające, czy agent odniósł sukces, czy też poniósł klęskę

Środowisko zadania (2)

- Niech $\mathscr{R}_{\Psi}(Ag, Env)$ oznacza zbiór wszystkich przebiegów agenta Ag w środowisku Env, które spełniają Ψ

$$\mathscr{R}_{\Psi}(Ag, Env) = \{r \,|\, r \in \mathscr{R}(Ag, Env) \wedge \Psi(r) = 1\}$$

- Agent Ag odnosi sukces w środowisku zadaniowym $\langle Env, \Psi \rangle$ jeśli
  - każdy przebieg prowadzi do sukcesu (definicja pesymistyczna) – $\forall r \in \mathscr{R}(Ag, Env) : \Psi(r) = 1$ $(\mathscr{R}_{\Psi}(Ag, Env) = \mathscr{R}(Ag, Env))$
  - co najmniej jeden przebieg prowadzi do sukcesu (definicja optymistyczna) – $\exists r \in \mathscr{R}(Ag, Env) : \Psi(r) = 1$ $(\mathscr{R}_{\Psi}(Ag, Env) \neq \emptyset)$

Prawdopodobieństwo sukcesu

- Prawdopodobieństwo $P(\Psi \mid Ag, Env)$ spełniania Ψ przez agenta Ag w środowisku Env jest zdefiniowane jako

$$P(\Psi \mid Ag, Env) = \sum_{r \in \mathscr{R}_\Psi(Ag, Env)} P(r \mid Ag, Env)$$

- $P(\Psi \mid Ag, Env)$ warto rozważać dla optymistycznej definicji sukcesu – dla pesymistycznej zawsze równe 1.0

Zadania osiągnięcia i utrzymania

Dwa podstawowe typy zadań (defniowane poprzez predykatową specyfikację zadania):

1. zadania osiągnięcia (*achievement tasks*) $\rightarrow$ osiągnięcie pewnego stanu rzeczy ϕ
2. zadania utrzymania (*maintenance tasks*) $\rightarrow$ utrzymanie pewnego stanu rzeczy ψ

Zadanie osiągnięcia

- Zadanie osiągnięcia zdefiniowane jest poprzez zbiór $\mathscr{G}$ zawierający „dobre” lub „docelowe” stany środowiska ($\mathscr{G} \subseteq E$)
- Agent odnosi sukces, jeśli osiąga jeden (dowolny) ze stanów ze zbioru $\mathscr{G} \rightarrow$ gra przeciw środowisku
- $\langle Env, \mathscr{G} \rangle$ oznacza środowisko zadania osiągnięcia ze stanami docelowymi $\mathscr{G}$ i środowiskiem Env

Zadanie utrzymania

- Zadanie utrzymania jest zdefiniowane poprzez zbiór $\mathscr{B}$ zawierający „złe” stany (lub stany „porażki”) środowiska($\mathscr{B} \subseteq E$)
- Agent odnosi sukces jeśli uda mu się uniknąć wszystkich stanów ze zbioru $\mathscr{B}$, tzn. nigdy nie wykonuje żadnej akcji prowadzącej do stanu z $\mathscr{B} \rightarrow$ środowisko gra przeciwko agentowi
- $\langle Env, \mathscr{B} \rangle$ oznacza środowisko zadania utrzymania ze stanami porażki $\mathscr{B}$ i środowiskiem Env

Synteza agentów

- Synteza agentów to problem automatycznego programowania (również rozważany w ramach AI $\rightarrow$ trenowanie graczy komputerowych w grach)
- Cel: automatyczne stworzenie na podstawie środowiska zadania specyfikacji takiego agenta, który odniesie sukces w tym środowisku
- Formalnie, algorytm syntezy agenta *syn* może być reprezentowany przez następująca funkcję ($\perp$ oznacza `null`)

$$syn : \mathscr{TE} \rightarrow (\mathscr{AG} \cup \{\perp\})$$

Solidność i kompletność (1)

Algorytm syntezy *syn* jest

1. solidny (*sound*), jeśli każdy stworzony agent odnosi sukces we wskazanym środowisku zadania (agent nie zawsze musi zostać stworzony!)
2. kompletny (*complete*), jeśli gwarantuje stworzenie agenta, o ile tylko istnieje agent odnoszący sukces w środowisku zadania (stworzony agent nie musi odnosić sukcesu!)

Idealny algorytm syntezy powinien być solidny i kompletny, przy czym ta pierwsza cecha jest bardziej istotna

Solidność i kompletność (2)

Formalnie algorytm *syn* jest

- solidny, jeśli

$$syn(\langle Env, \Psi \rangle) = Ag \implies \mathscr{R}(Ag, Env) = \mathscr{R}_{\Psi}(Ag, Env)$$

- kompletny, jeśli

$$\exists Ag \in \mathscr{AG} : \mathscr{R}(Ag, Env) = \mathscr{R}_{\Psi}(Ag, Env) \implies syn(\langle Env, \Psi \rangle) \neq \bot$$