Ontologia na potrzeby sterowania robotem usługowym*

Cezary Zieliński[1], Tomasz Kornuta[1]

Streszczenie

Zadania robotów realizowane są w ich otoczeniu, natomiast sterowaniu podlegają efektory tych robotów. Jeżeli zadanie do zrealizowania przez robota ma być wyrażone poprzez pojęcia, którymi zwykliśmy się posługiwać opisując otoczenie, to musimy stworzyć algorytm transformacji tego opisu w operacje wykonywane przez robota. Model otoczenia zwykło się wyrażać jako zbiór obiektów egzystujących w środowisku oraz relacji pomiędzy tymi obiektami. Planowanie działań, będące działem sztucznej inteligencji, korzysta z podania sekwencji statycznych sytuacji, opisanych w kategoriach relacji miedzy obiektami, oraz operacji przekształcających jedną sytuację w drugą. Ta praca koncentruje się na formalnym związku między relacjami opisującymi zaistniałe sytuacje i operacjami, które jest w stanie zrealizować robot. Ponieważ dążymy do skonstruowania układu sterowania robota, któremu zadania będzie można formułować w kategoriach relacji między obiektami, to rozważania obejma również strukturę takiego układu sterowania.

1. WSTĘP

Zazwyczaj zadanie, które ma być zrealizowane, albo jest określane jako sekwencja czynności, które ma wykonać robot traktowany jako urządzenie, albo jako sekwencja czynności, które mają być wykonane w środowisku, a sam robot, pomimo że jest czynnikiem sprawczym, to pozostaje w tym opisie niewidoczny. Ten artykuł koncentruje się na obu sposobach opisu, a w szczególności na transformacji z opisu zorientowanego na środowisko do opisu zorientowanego na robota. Większość autorów zajmujących się określaniem zadania w środowisku definiuje je poprzez hierarchicznie określone czynności. Czynności bardziej złożone składają się z czynności prostszych (bardziej elementarnych). Obiekty znajdujące się w środowisku stanowią parametry tych czynności. W przedstawianej pracy przyjęto konwencję odwrotną – obiekty stanowią kategorie pierwotna, natomiast relacje są definiowane poprzez odwołanie się do klas obiektów i ich atrybutów. Czynności do wykonania wynikają z relacyjnego opisu sytuacji, które powinny zaistnieć w trakcie realizacji zadania.

Opis zadania do zrealizowania przez robota może polegać albo na określeniu wszystkich kolejnych czynności, które mają być wykonane (ang. *Task Scripting*), albo określeniu co ma być zrobione za pomocą pojęć ogólnych, z których automatycznie są generowane konkretne czynności do wykonania (ang. *Task Planning*) [2]. W tym drugim przypadku planer, na podstawie ogólnej specyfikacji zadania, generuje

*Praca wykonana w ramach grantu Narodowego Centrum Badań i Rozwoju numer PBS1/A3/8/2012.

[1]Instytut Automatyki i Informatyki Stosowanej, Politechnika Warszawska, ul. Nowowiejska 15/19, 00-665 Warszawa; {T.Kornuta,C.Zieliński}@ia.pw.edu.pl

sekwencję czynności. Tak więc w obu przypadkach mamy do czynienia ze stworzeniem pełnej sekwencji czynności elementarnych, ale w pierwszym przypadku czyni to programista, a w drugim planer. Wspomniane czynności zazwyczaj przypisywane są węzłom grafu reprezentującego automat skończony określający ich kolejność, a dla ułatwienia programowania korzysta się z hierarchicznych automatów skończonych – w tym przypadku węzły grafu automatu reprezentują albo pojedyncze czynności albo grafy hierarchicznie niższych automatów (np. SMACH [2] lub ROSCo [8]). Od powstania robotyki dążono do wyrażania zadań dla robotów w jak najprostszy sposób, a więc najlepiej w języku naturalnym. Niestety, język naturalny jest wieloznaczny i co więcej zakłada gigantyczną wiedzę ze strony odbiorcy przekazu, gdyż wszelkie niejednoznaczności bądź niedomówienia rozstrzygane mogę być jedynie na podstawie posiadanej wiedzy. Stąd też duże zainteresowanie językami pośrednimi, które są podobne do języków naturalnych, ale pozwalają w istotny sposób zredukować niejednoznaczność (np. [9]). Do rozstrzygania niedopowiedzeń służą bazy wiedzy (np. KnowRob [10, 11]). Ponieważ bazy te muszą zawierać bardzo dużo informacji obecne wysiłki skierowane są na ich automatycznie tworzenie, np. na podstawie wiedzy zawartej w WWW. Projekt RoboEarth ma za zadanie stworzenie sieci WWW dla robotów, do której jedne roboty będą mogły dostarczać zdobytą wiedzę, tak aby inne roboty mogły z niej korzystać [12]. Wspomniane projekty są nakierowane na tworzenie odpowiedniej reprezentacji wiedzy, jej automatyczne zbieranie i opracowanie metod generacji programów wynikowych w kategoriach zrozumiałych przez bezpośrednie sterowniki robotów. Niniejsza praca ma na celu zbadanie możliwości stworzenia warstwy pośredniej, a więc sposobu określenia zadania w kategoriach obiektów istniejących w środowisku, ale nie wymagających bazy wiedzy do wygenerowania zestawu czynności do wykonania przez robota.

W pracy podjęto próbę znalezienia relacji miedzy obiektami, o takiej złożoności ich definicji, że z jednej strony opis zadania nie będzie wymagał odwoływania się do robota bezpośrednio, a z drugiej strony będzie umożliwiał określenie parametrycznego algorytmu realizacji takiej relacji jako automatu skończonego wywołującego zachowania robota zdefiniowane w kategoriach funkcji przejścia oraz warunków początkowych i końcowych. Podjęto tu próbę sformułowania odpowiedzi na pytanie: „Jaki poziom ogólności specyfikacji zadania nie pociąga za sobą konieczności wykorzystania wnioskowania do jego wykonania?”. Skoncentrowano się na sformułowaniu ogólnego algorytmu prowadzącego do powstania pożądanej sytuacji, przy czym osiągniecie tej sytuacji nie powinno wymagać użycia wnioskowania. Oczywiście posiadanie przez robota umiejętności wnioskowania jest cechą pożądaną, ale należy pamiętać, że wnioskowanie wymaga z jednej strony zgromadzenia znacznych zasobów wiedzy dotyczącej obiektów znajdujących się w środowisku, a z drugiej strony wydajnych algorytmów przetwarzania tej wiedzy, w celu wyciągnięcia pożądanych wniosków. Formułowanie zadań dla robota w języku naturalnym człowieka niewątpliwie wymaga dużej dozy wnioskowania i planowania z jego strony. Przetwarzanie języka naturalnego oraz planowanie zadań są rozpatrywane jako oddzielne dziedziny i często nie odwołują się do rzeczywistych robotów, ale działają w sztucznie stworzonym świecie. Rozziew miedzy efektami pracy takich systemów a możliwościami robotów bywa znaczny. Niniejsza praca ma na celu zapełnić powstałą lukę.

Systemy robotyczne wykorzystują relacje między obiektami znajdującymi się

w otoczeniu robota do:

1. Opisu chwilowego stanu środowiska (opis statyczny, np. [5]),
2. Sprawdzenia, czy środowisko ma oczekiwany stan,
3. Wytworzenia pożądanego stanu środowiska (zmiana stanu, np. [14, 15]).

O ile do osiągnięcia dwóch pierwszych celów wystarczy odwołanie się do relacji geometrycznych, to w ostatnim przypadku już tak nie jest. Ten artykuł koncentruje się wytwarzaniu pożądanego stanu środowiska. Stan ten opisywany będzie za pomocą relacji miedzy obiektami, które wtórnie zostaną zdefiniowane w kategoriach relacji matematycznych między atrybutami tych obiektów.

2. MODEL ŚRODOWISKA

Proste roboty mogą korzystać ze sterowników reaktywnych, gdzie reakcje robota są bezpośrednio wywoływane bodźcami uzyskiwanymi dzięki jego czujnikom [1, 3, 4]. W przypadkach bardziej złożonych potrzebny jest model środowiska, aby móc planować i wnioskować o rezultatach wykonywanych przez robota operacji. Oczywiście planowanie nie wyklucza współpracy z reaktywną warstwą sterowania – istnieją więc także systemy hybrydowe [1]. Należy też zwrócić uwagę, że pojęcia użyte do skonstruowania modelu tworzą ontologię, którą posługuje się nie tylko robot, ale również użytkownik formułujący zadanie do wykonania przez tego robota [5, 14]. Aby opis zadania dostarczany robotowi przez użytkownika można było przełożyć na sekwencje operacji, które robot wykona, zarówno sam opis zadania, jak i definicje operacji, które robot jest w stanie wykonać, trzeba sformalizować.

Fizycznemu środowisku rzeczywistemu $\mathcal{E}^*$ odpowiada model $\mathcal{E}$, którym posługuje się zarówno robot, jak i użytkownik. W środowisku rzeczywistym $\mathcal{E}^*$ znajdujące się rzeczywiste obiekty o_ν^*, $\nu = 1, \ldots, n_o$. Obiektom rzeczywistym o_ν^* odpowiadają ich modele o_ν. n_o jest liczbą obiektów w środowisku i w jego modelu. Liczby obiektów w środowisku rzeczywistym i w jego modelu są takie same, gdyż obiekty, które nie są modelowane z punktu widzenia robota nie istnieją. Modele obiektów podzielone są na klasy O_ξ, $\xi = 1, \ldots, n_c$, gdzie n_c jest liczbą klas obiektów. Zbiór atrybutów związanych z klasą obiektów O_ξ reprezentowany jest przez $\mathcal{D}_\xi$. Zbiór wszystkich możliwych atrybutów to $\mathcal{D} = \bigcup_{\xi=1}^{n_c} \mathcal{D}_\xi$. Atrybut α klasy obiektów O_ξ reprezentowany jest przez $d_{\xi,\alpha}$. Klasy obiektów wprowadzono, aby móc pogrupować obiekty według wspólnych własności reprezentowanych atrybutami tego samego typu. Konkretne obiekty, przykładowo o_1 i o_2, tej samej klasy mają atrybuty $d_{\xi,\alpha}$ tego samego typu, ale ich wartości mogą być różne i zazwyczaj sa różne. Relacje obiektowe definiowane są dla klas obiektów, natomiast zachodzą lub są tworzone dla konkretnych obiektów tych klas. Relacje obiektowe można zdefiniować w kategoriach matematycznych relacji między atrybutami, co w konsekwencji umożliwia ich transformację w sekwencje operacji, które robot musi wykonać, aby taka relacja zaistniała. To, czy taka transformacja wymaga planowania i wykorzystania wnioskowania logicznego, czy też można podać sztywny algorytm przekształcenia tak zdefiniowanej relacji w sekwencję operacji stanowi otwarty problem badawczy. W tej pracy, na wzór [14, 15], poszukiwany jest algorytm, przypisania relacji obiektowej sekwencji operacji ją wytwarzających, bez wykorzystania metod stosowanych w sztucznej inteligencji.

Tutaj będą rozpatrywane jedynie unarne ${}^{u}\mathcal{R}_{\eta}$ i binarne ${}^{b}\mathcal{R}_{\eta}$ relacje obiektowe. Te pierwsze dotyczą cech pojedynczych obiektów, natomiast te drugie wiążą się z geometryczna lokalizacja dwóch obiektów względem siebie. Zbiór relacji obiektowych $\mathcal{R}_{\eta}$ obejmuje zarówno relacje unarne jak i binarne. n_{ρ_b} to liczba rozpatrywanych relacji obiektowych. Stosując zapis infiksowy relacja binarna ${}^{b}\mathcal{R}_{\eta}$ zachodząca między między obiektami o_{κ} i o_{λ} wyrażana jest jako:

$$o_{\kappa}\ {}^{b}\mathcal{R}_{\eta}\ o_{\lambda} \qquad \text{np.: Miska Na Stole} \tag{1}$$

natomiast relacje unarne będą przedstawiane w formie prefiksowej:

$${}^{u}\mathcal{R}_{\eta}\ o_{\kappa} \qquad \text{np.: Czerwona Miska} \tag{2}$$

3. DEFINICJE RELACJI

Istnieje wiele możliwych typów definicji relacji obiektowych. Można wykorzystać zarówno definicje równościowe, jak i nierównościowe.

Relacje binarne ${}^{b}\mathcal{R}_{\eta}$ definiowane są między dwoma klasami obiektów, a więc wykorzystuje się do tego celu atrybuty przypisane tym klasom. Definicje równościowe przyjmują postać:

$$F_{\eta,1}(d_{\kappa,\alpha}) = F_{\eta,2}(d_{\lambda,\beta}), d_{\kappa,\alpha} \in \mathcal{D}_{\kappa}, \quad d_{\lambda,\beta} \in \mathcal{D}_{\lambda} \tag{3}$$

natomiast definicje nierównościowe:

$$F_{\eta,2}(d_{\lambda,\beta}) \leq F_{\eta,1}(d_{\kappa,\alpha}) \leq F_{\eta,3}(d_{\lambda,\beta}) \tag{4}$$

gdzie $F_{\eta,\zeta}, \zeta = 1,2,3$, to odpowiednio zdefiniowane funkcje.

Obiektowe relacje unarne ${}^{u}\mathcal{R}_{\eta}$ określają cechy obiektów, a więc ich definicje ustalają wartości atrybutów pojedynczego obiektu w relacji do stałych. Tu ponownie można skorzystać z definicji równosciowej

$$F_{\eta}(d_{\kappa,\alpha}) = \text{const}, \quad d_{\kappa,\alpha} \in \mathcal{D}_{\kappa} \tag{5}$$

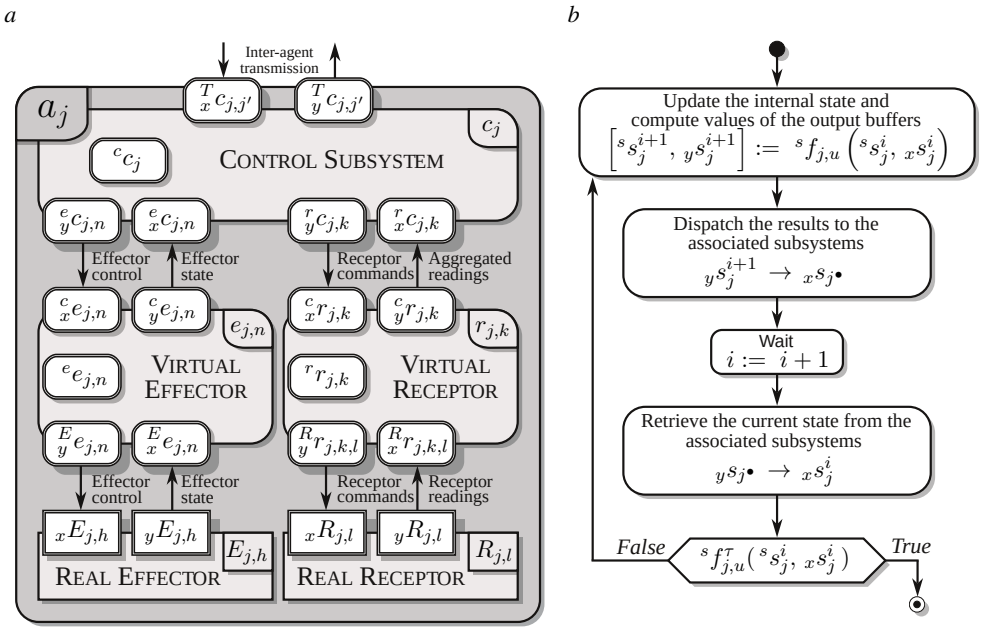
lub nierównościowej:

$$\text{const}_1 \leq F_{\eta}(d_{\kappa,\alpha}) \leq \text{const}_2 \tag{6}$$

gdzie F_{η} to odpowiednio zdefiniowana funkcja.

4. ROBOT

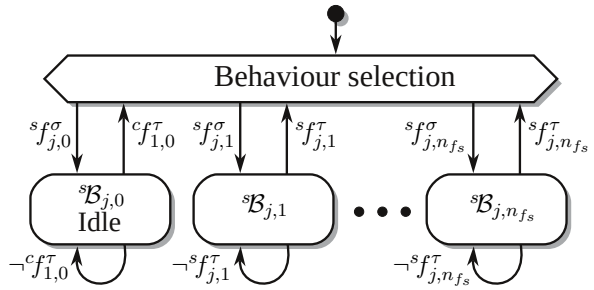
Robot reprezentowany jest przez agenta upostaciowionego [6, 7, 13, 16]. W tym artykule, dla uproszczenia, zakładamy istnienie jednego agenta, ale w ogólności takich agentów może być więcej, stąd agenty reprezentowane są przez a_j, $j = 1, \ldots, n_a$, gdzie



Rys. 1. a – Wewnętrzna struktura agenta a_j b – Sieć działań zachowania ${}^{s}\mathcal{B}_{j,u}$

n_a jest ich liczbą. Indeks j wskazuje konkretnego agenta i może przyjąć postać zarówno liczbową jak i symboliczną. Agent a_j w swej najogólniejszej postaci posiada efektory E_j, które wpływają na stan otoczenia, receptory R_j (a dokładniej eksteroreceptory), które pozyskują informacje o stanie otoczenia, oraz układ sterowania C_j, który steruje pracą agenta. Eksteroreceptory agenta a_j są albo numerowane albo nazywane, stąd $R_{j,l}$, $l = 1,\ldots,n_R$. Dotyczy to także efektorów $E_{j,h}$, $h = 1,\ldots,n_E$. Zarówno odczyty receptorów jak i polecenia dla efektorów muszą być reprezentowane w formie umożliwiającej łatwe przekształcenia w układzie sterowania, w związku z czym układ sterowania został zdekomponowany na wirtualne receptory $r_{j,k}$, $k = 1,\ldots,n_r$, wirtualne efektory $e_{j,n}$, $n = 1,\ldots,n_e$, oraz podsystem sterowania c_j (rys. 1a). Wirtualne receptory odpowiedzialne sa za agregację danych pomiarowych otrzymanych z rzeczywistych receptorów $R_{j,l}$ do postaci akceptowanej przez podsystem sterowania, natomiast wirtualne efektory przekształcają polecenia podsystemu sterowania do postaci akceptowalnej dla rzeczywistych efektorów $E_{j,h}$. Podsystem sterowania musi być w stanie zarówno rekonfigurować receptory $R_{j,l}$ jak i pozyskiwać informację proprioreceptywną od efektorów $E_{j,h}$, a więc muszą również istnieć przepływy wsteczne poprzez wirtualne receptory i efektory. Podsystem sterowania c_j umożliwia również komunikację agenta a_j z innymi $a_{j'}$, $j \neq j'$.

Wymienione elementy układu sterowania C_j wymieniają informacje za pomocą buforów (rys. 1a). W dalszej części artykułu zastosowano następującą konwencje notacyjną. Wprawdzie nie rozróżniono nazwy elementu i jego stanu, ale kontekst czyni to rozróżnienie oczywistym. W przyjętej notacji jednoliterowy symbol umieszczony w centrum (czyli E, R, e, r, c wskazuje rozważany element. Aby rozróżnić



Rys. 2. Graf automatu skończonego wybierającego zachowania

inne elementy tego typu, wskazać role danego tworu, bądź wyznaczyć chwilę, w której rozpatrujemy stan tego elementu umies‘zczono indeksy wokół symbolu centralnego. Indeks umieszczony w lewym górnym rogu wskazuje element, do którego bufor jest dołączony. Prawy górny indeks determinuje dyskretny czas. Lewy dolny indeks wskazuje typ bufora (x – wejściowy, y – wyjściowy, a jego brak – pamięć wewnętrzną). Indeksy umieszczone z prawej u dołu kolejno wskazują: agenta i komponent danego typu. Przykładowo, ${}^{c}_{x}e^{i}_{j,n}$ jest zawartością bufora wejściowego $n$tego efektora wirtualnego agenta a_j w chwili i, uzyskującego dane od podsystemu sterowania c_j.

Cykl pracy podsystemu s (gdzie $s \in \{c,e,r\}$) agenta a_j przedstawiono na rys. 1b. Funkcjonowanie elementu s określone jest za pomocą funkcji przejścia, której argumentami są dane zawarte w jego buforach wejściowych ${}_{x}s_j$ i pamięci wewnętrznej ${}^{s}s_j$, natomiast funkcja wytwarza wartości dla buforów wyjściowych ${}_{y}s_j$ i uaktualnia stan pamięci wewnętrznej ${}^{s}s_j$. Zachowanie elementu definiowane jest poprzez podanie funkcji ${}^{s}f_j$:

$$\left[{}^{s}s^{i+1}_{j}, {}_{y}s^{i+1}_{j}\right] := {}^{s}f_j({}^{s}s^{i}_{j}, {}_{x}s^{i}_{j}). \tag{7}$$

gdzie i oraz $i+1$ są kolejnymi chwilami. Funkcja (7) opisuje ewolucję stanu elementu s. Pojedyncza funkcja (7) byłaby bardzo złożona, dlatego podlega dekompozycji:

$$\left[{}^{s}s^{i+1}_{j}, {}_{y}s^{i+1}_{j}\right] := {}^{s}f_{j,q}({}^{s}s^{i}_{j}, {}_{x}s^{i}_{j}), \tag{8}$$

gdzie $q = 0,\ldots,n_{f_s}$. Dekompozycja implikuje wybór funkcji w czasie pracy systemu oraz wskazanie momentu, w którym dana funkcja ma przerwać swoje działanie. Do tego celu służą dwa predykaty:

- ${}^{s}f^{\sigma}$ określająca warunek początkowy zezwalający na wykorzystanie danej funkcji przejścia oraz
- ${}^{s}f^{\tau}$ definiująca warunek końcowy determinujący zakończenie pracy tej funkcji przejścia.

Tak więc wielokrokowa ewolucja stanu elementu definiowana jest poprzez zachowanie $\mathcal{B}_{j,u}$ sparametryzowane funkcją przejścia oraz warunkiem końcowym:

$${}^{s}\mathcal{B}_{j,u} \triangleq {}^{s}\mathcal{B}_{j,u}\left({}^{s}f_{j,u}, {}^{s}f^{\tau}_{j,u}\right) \tag{9}$$

Symbol $s_{j\bullet}$, gdzie $j^{\bullet} \in \{j, j'\}$, reprezentuje dowolny element połączony z s_j (w przypadku podsystemu sterowania element ten może nie należeć do rozpatrywanego agenta, stąd pojawiło się j'). Wybór tego zachowania determinowany jest warunkiem początkowym, który etykietuje łuki grafu automatu skończonego realizującego zadanie stanowiące imperatyw dla agenta (rys. 2). Zachowania ${}^{s}\mathcal{B}_{j,u}$ przyporządkowywane są węzłom grafu automatu. ${}^{s}\mathcal{B}_{j,0}$ jest zachowaniem biernym, aktywowanym gdy żadne inne zachowanie nie może być zrealizowane. Wybór zachowania reprezentowany na rys. 2 blokiem sześciokątnym wykonywany jest jako przełączenie bezstanowe – definiuje tablicę przejść automatu określona za pomocą warunków początkowych ${}^{s}f^{\sigma}_{j,u}$.

5. REALIZACJA RELACJI

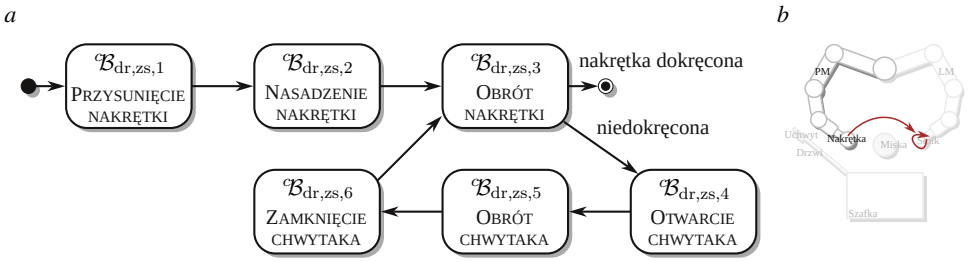
Relacje między obiektami opisują stan środowiska w danej chwili. Sekwencja takich opisów (sytuacji) może reprezentować plan działań robota. Jeżeli między kolejnymi sytuacjami pojawia się tylko jedna nowa relacja i zrywana jest co najwyżej jedna istniejąca relacja, to można określić algorytm przejścia z jednej sytuacji do drugiej w postaci grafu automatu skończonego, do węzłów którego przyporządkowano zachowania zdefiniowane w kategoriach funkcji przejścia oraz warunków końcowych i początkowych. W literaturze (np. [11]) często opis relacyjny uzupełniany jest podaniem konkretnych czynności, które prowadzą od sytuacji początkowej do kolejnej w sekwencji. Czy taki dodatkowy opis determinujący czynności robota, a więc jego zachowania, jest niezbędny, jest otwartym problemem badawczym.

6. DEFINICJA ZADANIA JAKO SEKWENCJI RELACJI

Rozważane zadanie robota usługowego polega na wyjęciu z szafki zakręconego słoika, wsypanie jego zawartości do miski, a następnie odłożenie słoika na miejsce. Założono, że drzwi szafki na początku są zamknięte.

1. Pozycja szafki jest znana (Location? Cupboard),
2. Pozycja uchwytu drzwi szafki jest znana (Location? Door_handle),
3. Uchwyt drzwi jest uchwycony: (Door_handle Grasped Gripper),
4. Drzwi szafki są otwarte: (Door At Open_pose),
5. Pozycja słoika jest znana (Location? Jar),
6. Słoik jest uchwycony: (Jar Grasped Gripper),
7. Słoik na zewnątrz szafki (Jar At Outside_pose),
8. Słoik otwarty (Lid At Above_jar_pose),
9. Pozycja miski jest znana (Location? Bowl),
10. Zawartość słoika jest w misce (Jar_contents In Bowl),
11. Pokrywka jest na słoiku (Lid Twisted Jar),
12. Słoik jest w szafce (Jar On Cupboard_shelf),
13. Uchwyt drzwi jest uchwycony: (Door_handle Grasped Gripper),
14. Drzwi szafki są zamknięte: (Door At Closed_pose).

Zdefiniowanie wszystkich powyższych relacji w ramach krótkiego artykułu niestety nie jest możliwe, dlatego do ilustracji wywodu wybrano relację binarną Twisted. W



Rys. 3. a – Graf automatu skończonego realizującego relację Twisted b – Przykładowa realizacja relacji Twisted

powyższym przykładzie wykorzystano relację unarną Location do określenia pozycji danego obiektu. Pozycja ta ustalana jest poprzez układ percepcyjny robota, a więc do tego celu będą zaangażowane receptory. Znak zapytania podkreśla, że chodzi o ustalenie cechy obiektu, a nie nadanie jej konkretnej wartości.

7. PRZYKŁAD: REALIZACJA RELACJI: (Lid Twisted Jar)

Zakładając, że słoik Jar należy do klasy Jars (Jar $\in$ Jars), natomiast pokrywka od słoika lid należy do klasy Lids (Lid $\in$ Lids) i wiedząc, że klasy Jars i Lids posiadają następujące atrybuty:

$$\mathcal{D}_{\text{Jars}} = \{d_{\text{Jars,base}}, d_{\text{Jars,top}}, d_{\text{Jars,bottom}}, d_{\text{Jars,twist}}, d_{\text{Jars,grasped}}, d_{\text{Jars,handle}}, \ldots\} \quad (10)$$

$$\mathcal{D}_{\text{Lids}} = \{d_{\text{Lids,base}}, d_{\text{Lids,top}}, d_{\text{Lids,bottom}}, d_{\text{Lids,twist}}, d_{\text{Lids,grasped}}, \ldots\} \quad (11)$$

można zdefiniować relację Twisted pomiędzy obiektami klas Jars i Lids jako:

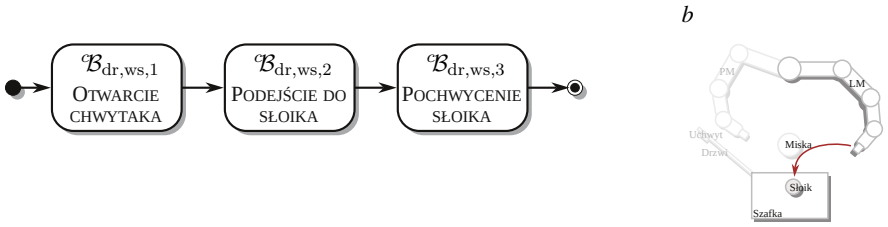
$$d^{\kappa+1}_{\text{Lids,base}} * d_{\text{Lids,bottom}} = d^{\kappa+1}_{\text{Jars,base}} * d_{\text{Jars,top}} \quad (12)$$

gdzie: $d_{\text{Lids,base}}$ i $d_{\text{Jars,base}}$ są macierzami jednorodnymi reprezentującymi pozycję układów bazowych odpowiednio pokrywki i słoika; $d_{\text{Lids,bottom}}$ i $d_{\text{Jars,top}}$ są macierzami jednorodnymi reprezentującymi pozycję spodu pokrywki i wierzchu słoika względem odpowiednich układów bazowych; $*$ to mnożenie macierzowe. Z kolei $\kappa+1$ oznacza chwilę czasową po wykonaniu czynności, a więc stan pożądany. Automat skończony realizujący relację (lid Twisted jar) przedstawiono na rys. 3a.

Oczywiście do realizacji niezbędny jest odpowiedni stan środowiska, który w tym przypadku zdefiniowaliśmy jako:

$$d_{\text{Lids,twist}} = (\text{negative\_thread}, \rho)$$
$$d_{\text{Jars,twist}} = (\text{positive\_thread}, \rho)$$
$$d^{\kappa}_{\text{Lids,grasped}} = \text{yes}$$
$$d^{\kappa}_{\text{Jars,grasped}} = \text{yes}$$

gdzie: $d_{\text{Lids,twist}}$ i $d_{\text{Jars,twist}}$ określają rodzaje gwintów i średnice pokrywki i słoiki – muszą one pasować do siebie, aby móc je ze sobą skręcić – w tym celu ρ określa



Rys. 4. a – Graf automatu skończonego realizującego relację Grasped b – Przykładowa realizacja relacji Grasped

jednakową średnicę słoika i pokrywki; $d_{\mathrm{Lids,grasped}}$ i $d_{\mathrm{Jars,grasped}}$ stwierdzają, że obiekty muszą być uchwycone, by można było wykonać operację prowadzącą do stanu, w którym pokrywka jest nakręcona na słoik. Pierwsza i druga linia definicji stwierdzają, że pokrywka i słoik muszą mieć jednakowe średnice oraz gwinty muszą do siebie pasować. Jeżeli tak jest, to wtedy należy doprowadzić do sytuacji, w której spód pokrywki będzie się pokrywał z wierzchem słoika. Ponieważ pokrywka i słoik posiadają pasujące gwinty, to należy doprowadzić do zdefiniowanej relacji przestrzennej poprzez nakręcanie, a nie nasuwanie. Ostatnie dwie linie definicji mówią, że pokrywka i słoik muszą być uchwycone przed rozpoczęciem realizacji relacji Twisted. Musi więc wpierw zaistnieć relacja Grasped, za realizację której odpowiedzialny jest automat skończony przedstawiony na rys. 4a. Tutaj ponownie pojawia się kwestia warunków, które muszą być spełnione, aby móc daną relację zrealizować, m.in. system musi znać pozycję chwytanego obiektu, co z kolei może wymagać wykorzystania percepcji w sposób aktywny w celu zlokalizowania tego obiektu.

Należy zwrócić uwagę, ze w proponowanej metodzie czynności prowadzące do powstania pożądanej relacji są wtórne – wynikają one z wartości atrybutów obiektów. To, że będzie stosowane nakręcanie wynika z faktu, iż słoik i pokrywka posiadają zdolność (ang. *affordance*) do bycia skręcanymi. Czy takie podejście, w opozycji do jawnego łączenia z relacją czynności, która prowadzi do jej powstania, będzie użyteczniejsze, będzie przedmiotem dalszych badań.

8. PODSUMOWANIE

W artykule uwaga skupiona na formalnym związku między relacjami opisującymi zaistniałe sytuacje i operacjami, które jest w stanie zrealizować robot. Mimo, iż cel nie został osiągnięty, wyniki wskazują, iż obrano właściwy kierunek badań. Dążymy bowiem do stworzenia takiego opisu, w którym plan działań robota będzie formułowany w kategoriach relacji dot. obiektów, ale nie będzie wymagał wykorzystania mechanizmów wnioskowania logicznego. W konsekwencji uzyskane zostaną elementarne zachowania wyrażane w kategoriach obiektów i relacji.

LITERATURA

[1] R. C. Arkin. *Behavior-Based Robotics*. MIT Press 1998.

[2] J. Boren, S. Cousins. The SMACH high-level executive. IEEE Robotics and Automation Magazine, December, 2010, s. 18–20.
[3] R. A. Brooks. Intelligence without reason. Artificial intelligence: critical concepts, 1991, wolumen 3, s. 107–63.
[4] R. A. Brooks. Intelligence without representation. Artificial Intelligence, January, 1991, wolumen 47, numer 1-3, s. 139–159.
[5] M. Fadarewski. *Inteligencja wokół nas. Współdziałanie agentów softwareowych, robotów, inteligentnych urządzeń*. EXIT 2010, wolumen 15, rozdział Reprezentacja środowiska robota, s. 13–42.
[6] T. Kornuta, C. Zieliński. Projektowanie układów sterowania robotów. część i: Metodyka. In: XII Krajowa Konferencja Robotyki – Postępy Robotyki. *Proceedings*. Oficyna Wydawnicza Politechniki Warszawskiej, 2012. wolumen 2 serii *Prace Naukowe – Elektronika*, s. 599–606.
[7] T. Kornuta, C. Zieliński. Robot control system design exemplified by multi-camera visual servoing. Journal of Intelligent & Robotic Systems, 2013, s. 1–25.
[8] H. Nguyen et al. Ros commander (rosco): Behavior creation for home robots. In: *IEEE International Conference on Robotics and Automation*. May, 2013.
[9] M. Stenmark, P. Nugues. Natural language programming of industrial robots. In: *44th International Symposium on Robotics, 24-26 October 2013, Seoul, Korea*. 2013.
[10] M. Stenmark, A. Stolt. A system for high-level task specification using complex sensor-based skills. In: *Robotics: Science and Systems (RSS) 2013 Workshop on Programming with Constraints, 2013-06-28, Berlin, Germany*. 2013.
[11] M. Tenorth, M. Beetz. KnowRob: a knowledge processing infrastructure for cognition-enabled robots. International Journal of Robotics Research, May, 2013, wolumen 32, numer 5, s. 566–590.
[12] M. Tenorth et al. Representation and exchange of knowledge about actions, objects, and environments in the RoboEarth framework. IEEE Transactions on Automation Science and Engineering, July, 2013, wolumen 10, s. 643 — 651.
[13] C. Zieliński, T. Kornuta. Diagnostic requirements in multi-robot systems. In: *Intelligent Systems in Technical and Medical Diagnostics*, s. 345–356. Springer 2014.
[14] C. Zieliński. TORBOL: An object level robot programming language. Mechatronics, 1991, Vol. 1, No. 4, s. 469–485.
[15] C. Zieliński. Description of semantics of robot programming languages. Mechatronics, 1992, Vol. 2, No. 2, s. 171–198.
[16] C. Zieliński, T. Kornuta, M. Boryń. Specification of robotic systems on an example of visual servoing. In: 10th International IFAC Symposium on Robot Control (SYROCO 2012). *Proceedings*, 2012. wolumen 10, s. 45–50.

ONTOLOGY FOR THE PURPOSE OF CONTROL OF A SERVICE ROBOT

The paper focuses on the relationship between objects and operations realized by a robot. As we strive to construct a robot control system, which tasks will be formulated in terms of relationships between objects, we consider both the structure of a control system, its behaviours and relations between objects of manipulation. The proposed approach express the elementary behaviours of a robotic system in terms of evolution of relationships between objects.