STUDIA METODOLOGICZNE
NR 34 • 2015

PAWEŁ GRABARCZYK

Trzy rodzaje reprezentacji w filozofii informatyki

1. Można odnieść wrażenie, że nawet jeśli pojęcie reprezentacji nie odgrywa w filozofii informatyki kluczowej roli, to stanowi swoistą „szarą eminencję". Pojęcie to wykorzystywane bywa bowiem często w rozważaniach dotyczących fundamentalnych zagadnień tej dziedziny, takich jak pytanie o naturę procesów obliczeniowych czy relacji zachodzących między oprogramowaniem i sprzętem. „Nie ma obliczania bez reprezentowania", jak poucza nas dobrze znany slogan Jerry'ego Fodora [Fodor, 1981]. Choć obliczanie polega jedynie na formalnych przekształceniach nośników treści, to miarą jego powodzenia ma być rzekomo to, czy treść wyjdzie z tego procesu bez szwanku. Zgodnie z tym obrazem komputery i inne maszyny liczące niczego innego nie robią, jak tylko systematycznie szeregują napływający do nich korowód reprezentacji. Fodor nie jest wcale w tym poglądzie odosobniony. Jak zauważa Vincent C. Müller [2008], wielu badaczy traktuje reprezentacje jako konieczny składnik teorii obliczeniowych (np. O'Brien, 1998, który wprowadza pojęcie reprezentacji do definicji obliczania). Analogiczne poglądy prezentują: Zenon W. Pylyshyn [1984] i Brian Cantwell Smith [2002]. Jako że, jak zobaczymy w dalszej części wywodu, najczęściej przywoływaną formą reprezentacji są reprezentacje symboliczne, to do autorów uznających reprezentacje za podstawę procesów obliczeniowych dołączyć możemy również Alana Turinga [Turing, 1950] i Allana Newella [1980].

Nie wszystkich takie postawienie sprawy zadowala. Niektórzy odwracają fodorowską zależność, twierdząc, że nie ma reprezentowania bez obliczania [Miłkowski, 2013]. Inni proponują ostateczne zerwanie z pojęciem reprezentacji w kontekście obliczania [Piccinini, 2008] lub alternatywne sposoby identyfikacji procesów obliczeniowych [Chalmers, 1996].

Problem natury procesu obliczania nie jest jednakże jedynym kontekstem filozofii informatyki, w którym non stop natykamy się na pojęcie reprezentacji. Wystarczy rzut oka w stronę badań nad Sztuczną Inteligencją, by to sobie uświadomić. Niektórzy [Müller, 2007] nazywają nawet reprezentacje centralnym pojęciem badań nad SI. Spowodowane to jest tym, że dla wielu badaczy zjawiska takie jak percepcja czy kategoryzacja w sposób konieczny wiążą się z pojęciem reprezentacji (czego często nawet się nie dowodzi). Przyjęcie tej perspektywy sprawia, że oczekujemy reprezentacji również w sztucznych ekwiwalentach tych procesów.

Badacze, którzy uważają, że o procesach tych mówić możemy bez powoływania się na reprezentacje, należą do rzadkości. Na dodatek ich poglądy stanowią zazwyczaj część większego i znacznie bardziej radykalnego pakietu poglądów [Brooks, 1991; Hutto, 2013], który pod znakiem zapytania stawia też tradycyjne rozumienie procesów obliczeniowych, a dla reprezentacji nie znajduje miejsca nie tylko w informatyce, ale i praktycznie nigdzie. Jeżeli bez reprezentacji nie ma obliczania, a bez obliczania – inteligencji, to tym gorzej dla reprezentacji – po prostu pojęcie to nie odegra żadnej roli w teoriach Sztucznej Inteligencji – zdaje się sugerować tytułem swojego artykułu Brooks [1991]. Autor ten idzie w swojej niechęci do reprezentacji tak daleko, że sugeruje nawet, nie bez złośliwości, że główną funkcją tego pojęcia jest powierzchowne spajanie zupełnie niepowiązanych ze sobą badań [Brookes, 1990].

Nie ulega zatem wątpliwości, że pojęcie „reprezentacji" pozostaje w centrum zainteresowania filozofów informatyki, nawet jeśli jest to negatywny punkt odniesienia [Müller, 2007]. Z jakiego powodu budzi ono takie kontrowersje?

2. Okazuje się, że mówiąc o „reprezentacjach", nietrudno wpaść w filozoficzne kłopoty. Można odnieść wrażenie, że pod względem niefraso-

bliwości i niestabilności użycia pojęcie to ustępuje jedynie pojęciu „informacji". Różnica polega jednak na tym, że o ile w przypadku tego drugiego możemy co jakiś czas przywołać się do porządku i cofnąć do dobrze zrozumiałej teorii Shannona [Shannon, 1948], to w przypadku reprezentacji po prostu nie ma gdzie się wycofać – nie dysponujemy bowiem żadną uznaną, precyzyjną jej teorią. Trudności związane z pojęciem reprezentacji są niezwykle różnorodne [Crane, 2003], sądzę jednak, że w kontekście filozofii informatyki szczególnie dotkliwe są trzy z nich.

Po pierwsze, pojęcie to nacechowane jest (prawdopodobnie nieusuwalną) wieloznacznością. Powiedzieć, że „tyle znaczeń, ilu autorów", byłoby zapewne przesadą, ale brak zgody co do tego, jak należy je rozumieć, przy jednoczesnej niechęci do podawania wyraźnej definicji, niezwykle utrudnia dyskusję, na co zresztą wielu badaczy się skarży [Morris, 1991]. Jest to szczególnie dotkliwe, ponieważ, jak zauważono już w literaturze, trzy wyraźnie odmienne sposoby rozumienia pojęcia reprezentacji daje się wskazać już od czasów Charlesa Sandersa Peirce'a, który, co w kontekście tego artykułu szczególnie istotne, stosował je do rozumowań matematycznych [Campos, 2009]. Prosta, lecz niezwykle poręczna klasyfikacja Peirce'a wyróżnia trzy typy reprezentacji: ikoniczne, wskaźnikowe oraz symboliczne. Pierwszy z nich łączy z reprezentowanym obiektem jakiegoś rodzaju podobieństwo, drugi – związek przyczynowo-skutkowy, trzeci – konwencja nadana przez użytkowników symboli. Jak zobaczymy w sekcji 3, podział ten okaże się zaskakująco użyteczny na gruncie filozofii indormatyki.

Po drugie, odwołanie się do reprezentacji przekierowuje często dyskusję na tory, na których jeszcze łatwiej się wykoleić. Dla wielu badaczy nie można po prostu „być reprezentacją". Jest się co najwyżej reprezentacją *czegoś* dla *kogoś*. Pierwszy z argumentów tej relacji (bycie reprezentacją *czegoś*) natychmiast przywodzi na myśl relację odniesienia przedmiotowego, która mimo wielu lat badań nadal jest przez wielu uważana za dość niepokojącą. Nawet optymiści w kwestii ustalania odniesienia przyznają, że słabo poddaje się ono naturalizacji, ponieważ sprawia wrażenie niefizycznego oddziaływania na odległość [Smith, 1995]. Tym trudniejsze do przełknięcia jest to na gruncie filozofii informatyki, ponieważ wydaje się

zakładać, że pewne niefizyczne relacje zachodzące na odległość pomiędzy maszyną a przedmiotami, z którymi nigdy się ona nie zetknęła, determinują mimo to jej działania. Z tego powodu, choć wielu badaczy podstaw informatyki odwołuje się do relacji odniesienia [Shagrir, 2010, s. 383], pojawiają się też propozycje zgoła przeciwne [Müller, 2007; Piccinini, 2008]. Drugi argument tak rozumianej relacji reprezentacji (bycie reprezentacją dla kogoś) odsyła nas zaś do funkcji interpretatora. Podobnie jak w przypadku relacji odniesienia, problem z interpretacją polega na trudności z jej naturalizacją. W jakim sensie o fizycznym obiekcie moglibyśmy orzec, że jest interpretatorem? Czy powiedzenie, że komputer interpretuje instrukcje, jest jedynie metaforą oznaczającą, że działa zgodnie z tą instrukcją, czy też cichym postulowaniem homunculusa w procesorze?

Po trzecie, jeżeli pojęcie reprezentacji ma w ogóle mieć zastosowanie w filozofii informatyki, należy ustalić, czy ma ono na gruncie tej dziedziny wartość eksplanacyjną. Jak zauważa William M. Ramsey [2007], pojęcie reprezentacji należy do grupy pojęć szczególnie zagrożonych eksplanacyjną jałowością. Po pierwsze dlatego, że jest ono zakorzenione w mowie potocznej, przez co może zostać przez badaczy bezwiednie zaszczepione na gruncie teorii bez nadania mu precyzyjnego technicznego znaczenia. Po drugie, ponieważ jest ono w badaniach psychologów i filozofów używane od dawna, może zostać siłą nawyku przeniesione ze starszej teorii (gdzie miało funkcję eksplanacyjną) do nowszej (gdzie już jej nie posiada).

3. Celem tego artykułu jest zaradzenie wyżej wymienionym trudnościom poprzez: (1) wyróżnienie i oddzielenie od siebie tych znaczeń terminu „reprezentacja", które wykorzystywane są w filozofii informatyki, (2) ustalenie, które z nich wymuszają na nas postulowanie odniesienia przedmiotowego i interpretatora oraz (3) sprawdzenie, które ze znaczeń rzeczywiście wnoszą coś do dziedziny, a które mogłyby równie dobrze zostać z niej wyeliminowane. Pomoże nam w tym trychotomia Peirce'a – okaże się bowiem, że trafnie oddziela ona trzy typy reprezentacji, z którymi spotkać się można w filozofii informatyki[1]. Przyjrzyjmy się im dokładniej.

[1] Narzędzie to nie jest wykorzystywane w kontekście filozofii informatyki po raz pierwszy, patrz np. Müller, 2007; Steinem, 2013.

Jak wspomnieliśmy na początku, trzeci z wyróżnionych przez Peirce'a typów reprezentacji, czyli reprezentacje symboliczne, przywołuje się w kontekście filozofii informatyki najczęściej. Dlatego też od niego proponuję rozpocząć naszą analizę. Rozumowanie stojące za odwołaniem do tego typu reprezentacji streścić można następująco: komputery zajmują się obliczaniem, a obliczanie to procedura polegająca na przekształcaniu symboli za pomocą określonych reguł. Istotą symbolu jest to, że do czegoś się odnosi, coś reprezentuje. Gdyby nie reprezentował niczego, to praca maszyny byłaby w rzeczywistości jałowa – nieugruntowane w żaden sposób symbole nie wygenerują sensu same z siebie [Harnad, 1990; Searle, 1980]. A zatem, cała praca komputerów sprowadza się do manipulacji reprezentacjami. Na pierwszy rzut oka rzeczywiście wydaje się więc, że w informatyce nie sposób obejść się bez pojęcia reprezentacji symbolicznej. Mimo to będę argumentował za tym, że mimo pozoru intuicyjności rozumowanie to zawiera w sobie aż trzy poważne nieporozumienia.

Po pierwsze, nie powinniśmy zapominać o tym, że jeśli zajrzymy do wnętrza komputera, to jedynymi symbolami, jakie tam napotkamy, są oznaczenia chipów, którymi komputer ten manipulować nie potrafi. Komputery to fizyczne obiekty manipulujące napięciem elektrycznym, a nie symbolami. Jest co najwyżej tak, że fizykalny opis działającego komputera daje się przełożyć na formalny opis jakiegoś obliczenia [Searle, 1990]. Istnienie tego rodzaju przekładu pozwala nam wtedy powiedzieć, że komputer stanowi fizyczną *realizację* obliczenia albo że obliczenie zostało *zaimplementowane* w fizycznej strukturze komputera. Charakter i definicja relacji „implementacji" i „realizacji" należą do najbardziej kontrowersyjnych problemów w dziedzinie filozofii informatyki [patrz choćby: Chrisley, 1994; Chalmers, 1996; Copeland, 1996]. Stwierdzenie, że komputer „dokonuje operacji na symbolach", pomija całkowicie ten problem, przez co stanowi bardzo niebezpieczny skrót myślowy.

Po drugie, nawet jeśli przeniesiemy naszą uwagę z tego, co dzieje się w komputerze, na formalizm, którego działanie to jest realizacją, to nie powinniśmy zapominać, że odniesienia przedmiotowego jest w nim mniej

więcej tyle samo, co w niezinterpretowanym rachunku logicznym[2]. Mówienie o tym, co symbole takiego formalizmu reprezentują, jest więc po prostu nieporozumieniem. Co jest jego źródłem? Sądzę, że przyczyny mogą być wielorakie. Istnieje hipoteza, że w przypadku części badaczy odpowiedzialne są za to skojarzenia z terminologią używaną w języku LISP [Miłkowski, 2013]. Wydaje mi się jednak, że równą, a może i większą winę ponosi za to sposób, w jakich przybliżamy sobie języki programowania. Mówimy często, że dane symbole „reprezentują funkcje matematyczne albo logiczne". Należy sobie jednak uświadomić, że jest to jedynie szkodliwy sposób mówienia – w rzeczywistości nie mamy tu do czynienia z relacją odniesienia, a z relacją synonimiczności. Gdy mówię, że operacja AND w asemblerze reprezentuje koniunkcję, to znaczy to tylko tyle, że funkcjonuje ona w tym języku analogicznie do koniunkcji w logice. O żadnym odniesieniu, a więc i reprezentowaniu, nie ma tu wcale mowy. Ucząc języka programowania dziecko, które nie zna logiki i matematyki, możemy pominąć zupełnie tego rodzaju analogie i nic strasznego się nie stanie.

Po trzecie – innym szkodliwym sposobem mówienia, który sugeruje, że odniesienie przedmiotowe odgrywa istotną rolę w funkcjonowaniu komputerów, jest stwierdzenie, że zmienne reprezentują jakieś rzeczywiste wartości. Choć w tym przypadku mamy już do czynienia z odniesieniem przedmiotowym z prawdziwego zdarzenia, to nie ma wątpliwości co do tego, że jest ono najczęściej dostarczone komputerowi z zewnątrz – przez programistę lub użytkownika. Jako takie nie ma ono żadnego istotnego znaczenia dla rzeczywistego przebiegu programu i mogłoby być zupełnie dobrze całkowicie pominięte w teorii działania maszyny. Można zatem powiedzieć, że określenie „zmienna A odnosi się do wielkości X" można sparafrazować do postaci „Użytkownik wykorzystuje zmienną A do obliczania wielkości X". To, do czego program jest wykorzystywany, nie stanowi jednakże o jego tożsamości. Jeżeli ktoś weźmie program do oblicza-

[2] Pewną komplikację wprowadzają tu dodatkowe stałe, które mogą się w takim formalizmie pojawić, ale ich odniesienie wyjaśnić można w sposób, jaki omawiam w kolejnych akapitach (możliwe jest uznanie ich za reprezentacje wskaźnikowe).

nia liczby przeżytych dni i zamiast daty urodzenia wprowadzi datę ślubu, uzyskując w ten sposób wynik w postaci liczby dni, które upłynęły od ślubu, to nie zmienił tym samym programu i jest co najwyżej „hakerem intencjonalnym"[3].

W praktyce oznacza to, że pojęcie reprezentacji symbolicznej, choć tak często z filozofią informatyki wiązane, nie ma na gruncie tej teorii istotnego zastosowania. Jego użycie sprowadza się bowiem albo do niezręcznego sposobu mówienia, którego precyzacja nie zawiera już terminu „reprezentacja", albo do kontekstów pobocznych, takich jak nauka programowania czy sposób wykorzystania oprogramowania przez użytkownika, ale te konteksty nie wpływają w żaden sposób na problematykę, którą zajmuje się zazwyczaj filozofia informatyki.

Rezygnacja z pojęcia relacji symbolicznej sprawia, że nic nie zmusza nas już do przypisywania stanom wewnętrznym komputera odniesienia przedmiotowego[4]. Argument, że inaczej nigdy nie będą one mogły stać się nośnikami sensu, również nie jest tak wystarczająco silny, jak chcieliby tego jego orędownicy. Po pierwsze, w przypadku niektórych wyrażeń (np. wspomnianych spójników logicznych) nie ma chyba nic zdrożnego w uznaniu, że ich znaczeniem jest po prostu rola, jaką odgrywają w transformacjach zdań. Po drugie, w przypadku wielu innych wyrażeń wystarczy nam odniesienie do stanów wewnętrznych systemu, które to odniesienie daje się objaśnić w dość wygodny sposób dzięki drugiemu typowi reprezentacji – reprezentacjom wskaźnikowym. Przyjrzyjmy się im bliżej.

Relację reprezentacji wskaźnikowej rozumie się zazwyczaj jako stabilną korelację opartą na prawach przyrody [Fodor, 1998, s. 12; Dretske, 1988; Field, 1978]. Modelowym przykładem tego rodzaju reprezentacji są wszelkiego rodzaju naturalne oznaki – dym znamionujący ogień, ślady w lesie oznaczające obecność dzika itd. Reprezentacje rozumiane w ten sposób już na pierwszy rzut oka wydają się mniej kłopotliwe niż reprezentacje symboliczne – pozwalają one bowiem mówić o relacji odniesienia

[3] Jak zobaczymy poniżej, o „odniesieniu zmiennych" mówi się także w innym kontekście, ale daje się on wyjaśnić za pomocą pojęcia S-reprezentacji wtórnej.

[4] Przez „stany wewnętrzne" rozumiem tu (i w innych miejscach) fizyczne stany wewnętrzne maszyny, np. stan napięcia na tranzystorach.

bez potrzeby postulowania zewnętrznego interpretatora[5]. Klasycznym przykładem, którym posługuje się Dretske, są słoje w drzewie [Dretske, 1988]. Zawierają one informacje o wieku drzewa, nawet jeśli nie istnieje interpretator zdolny do jej odczytania. Podawane przez autorów przykłady nie kojarzą się z filozofią informatyki, ponieważ dotyczą zazwyczaj gatunków naturalnych, ale nie należy się tym sugerować. Wszak komputery prawa przyrody również obowiązują. Należy jednocześnie podkreślić, że mówiąc, że tak rozumiane reprezentacje nie wymagają interpretatora, mam na myśli jedynie to, że nie wymuszają one na nas postulowania świadomego agenta, który odczytywałby je za pomocą jakiejś znanej sobie konwencji. Chodzi jedynie o to, że związek pomiędzy nimi a obiektami, które reprezentują, nie wykracza poza zwykłe prawa przyrody przez co nie prowadzą one do postulowania wewnętrznego „homunculusa". Nie zmienia to faktu, że każda z takich korelacji może, ale nie musi być przez jakiś system poznawczy wykorzystana. Zrobienie użytku z reprezentacji wskaźnikowej nie jest jednakże (nawet potocznie) nazywane aktem interpretacji[6]. Tym właśnie różni się zauważenie, że gdzieś się pali, od odczytania sygnałów dymnych.

I rzeczywiście – zastosowanie reprezentacji wskaźnikowych na gruncie filozofii informatyki nietrudno znaleźć. Moglibyśmy na przykład powiedzieć, że dany rejestr pamięci reprezentuje temperaturę procesora, ponieważ jest on po prostu przyczynowo powiązany z czujnikiem temperatury. Mimo to reprezentacje tego rodzaju omawiane są na gruncie filozofii informatyki dość rzadko[7]. Sądzę, że głównym powodem znikomej popularności tego sposobu rozumienia reprezentacji jest przekonanie o ich wąskiej stosowalności. Zazwyczaj przywołuje się je bowiem jedynie w kontekście wąskiej klasy sensorów wewnętrznych w rodzaju czujników temperatury

[5] Dla uniknięcia nieporozumień dodajmy, że brak konieczności postulowania interpretatora to perspektywa współczesna, z którą Peirce z pewnością by się nie zgodził.

[6] Z tego powodu badacze, którzy zajmują się tym rodzajem reprezentacji, mówią częściej o „konsumencie", a nie o „interpretatorze" reprezentacji [Millikan, 2004].

[7] Wyjątki, które warto odnotować to choćby Müller [2007], który wskazuje na „bezpieczną" formę odniesienia przedmiotowego w postaci odniesienia do danych sensorów, Fresco [2010], który wyraźnie wskazuje na konieczność oddzielania odniesienia wewnętrznego i zewnętrznego oraz Piccinini i jego semantyka wewnętrzna [2008].

czy też detektora stanu baterii. Warto jednak zauważyć, że reprezentacje wskaźnikowe nie ograniczają się do tego rodzaju czujników. Nawet w przypadku systemów o bogatym i zróżnicowanym kontakcie z otoczeniem (takich jak roboty) możliwe jest stosowanie alternatywnego opisu, w którym mówienie o odniesieniu przedmiotowym zastąpimy jego wąskim, wewnętrznym ekwiwalentem, który nie generuje trudnych pytań. Na przykład – zamiast mówić, że dany stan wewnętrzny reprezentuje dla systemu kolor czerwony, możemy mówić, że reprezentuje on pewien szczególny stan naświetlenia matrycy w kamerze (ten, który pojawia się jedynie w kontakcie z kolorem czerwonym). Odniesienie do przedmiotów zewnętrznych zostaje w ten sposób zastąpione znacznie mniej tajemniczym odniesieniem do rozmaitych wzorców pobudzeń receptorów. Reprezentacje tego typu zachowują więc jakiś ekwiwalent odniesienia przedmiotowego, ale relacja ta nie prowadzi do kłopotliwych konsekwencji, ponieważ daje się ją opisać w terminach fizykalnych, nie zachodzi na odległość i nie wymaga interpretatora.

Jak widzieliśmy w sekcji 2, Peirce, z którego typologii korzystamy, wyróżnił jeszcze trzecią klasę reprezentacji – reprezentacje ikoniczne. Czy znajdują one na gruncie filozofii informatyki jakieś zastosowanie? Współczesnym odpowiednikiem reprezentacji ikonicznych są reprezentacje strukturalne (zwane często dla uproszczenia S-reprezentacjami), omawiane choćby w Sheppard, Chapman, 1970; Cummins, 1989; Swoyer, 1991; Ramsey, 2007. Ten typ reprezentacji jest stosunkowo najrzadziej przywoływany (nawet na gruncie kognitywistyki), dlatego też proponuję przyjrzeć się mu nieco bliżej. S-reprezentacje definiowane są najczęściej w następujący sposób: o obiekcie A możemy powiedzieć, że reprezentuje obiekt B wtedy, gdy istnieje homomorfizm pozwalający na odwzorowanie relacji zachodzących w A na relacje zachodzące w obiekcie B. Paradygmatycznym przykładem S-reprezentacji są mapy. Nie muszą być one wizualnie podobne do odwzorowywanego terenu. Wystarczy, że sieć relacji pomiędzy elementami mapy (na przykład oznaczonymi na niej miastami albo ważnymi punktami orientacyjnymi) jest homomorficzna względem sieci relacji pomiędzy położeniem owych miast i punktów orientacyjnych. A zatem, choć odległości pomiędzy elementami nie będą raczej równe

odległościom pomiędzy miastami[8], to proporcje pomiędzy nimi będą równe proporcjom rzeczywistym.

Jak łatwo zauważyć, S-reprezentacje różnią się w sposób istotny od reprezentacji symbolicznych i interpretacji wskaźnikowych. Po pierwsze, relacja, która zachodzi pomiędzy nimi i pomiędzy reprezentowanymi obiektami, nie jest wynikiem żadnej konwencji (jak w przypadku reprezentacji symbolicznych), dlatego też nie wymagają one udziału żadnego interpretatora[9]. W odróżnieniu od symboli pozwalają one na identyfikację reprezentowanych obiektów na podstawie analizy samej ich struktury. Po drugie, w odróżnieniu od reprezentacji wskaźnikowych, nie są one z reprezentowanymi przez siebie obiektami powiązane przyczynowo. Z tego powodu relacja S-reprezentacji nie jest po prostu relacją odniesienia przedmiotowego (co bywa często wytykane jako wada tych reprezentacji). Zawsze może się tak zdarzyć, że mapa pasuje dość dobrze do kilku terenów. Do wyobrażenia sobie takiej sytuacji nie potrzebujemy nawet wytężać specjalnie wyobraźni. Mapa danego budynku jest równocześnie mapą każdego innego budynku wybudowanego według tego samego planu. Aby relację S-reprezentacji przekształcić w relację odniesienia, potrzebne jest spełnienie jakichś dodatkowych warunków a ich dobór jest przyczyną wielu sporów na gruncie kognitywistyki[10]. Nie trzeba chyba dodawać, że dla nas brak odniesienia to doskonała wiadomość, ponieważ automatycznie chroni nas przed problemami, które relacja ta rodzi. Nie determinując odniesienia i nie wymagając interpretatora, S-reprezentacje z całą pewnością stanowią filozoficznie najbezpieczniejszy typ reprezentacji. Mimo to z jakiegoś powodu bardzo rzadko dyskutuje się o tego typu reprezentacjach w kontek-

[8] Chyba że jest to mapa tak dokładna, że pokrywa po prostu cały teren [Borgis, 1998].

[9] Podobnie jak w przypadku reprezentacji wskaźnikowych nie oznacza to, że nie wymagają żadnego konsumenta. Chodzi jedynie o to, że posłużenie się S-reprezentacją nie jest aktem interpretacji rozumianej jako świadome posłużenie się konwencją i polegać może na przykład na zauważeniu podobieństwa struktury geometrycznej.

[10] Popularnym rozwiązaniem tego problemu jest odwołanie się do mechanizmu ewolucyjnego, w wyniku którego dana S-reprezentacja znalazła się w systemie [Millikan, 1984], co z oczywistych względów wyklucza zastosowanie na gruncie filozofii informatyki. Niektórzy badacze rozszerzają jednakże tę ideę na artefakty – decydujący jest wtedy powód, dla którego konstruktor wprowadził reprezentację do systemu [Dretske, 1995].

ście filozofii informatyki. Czy oznacza to, że są one na gruncie tej dziedziny eksplanacyjnie jałowe?

Do ustalenia tej kwestii potrzebne nam będzie dodatkowe narzędzie pojęciowe, a mianowicie rozróżnienie na model i symulację. Choć niektórzy autorzy posługują się tymi terminami zamiennie [Ramsey 2007], to jedna z różnic pomiędzy nimi będzie dla naszych rozważań istotna. Gdy mówimy, że X jest symulacją procesu A, to mamy na myśli, że są one pod istotnym względem równoważne, a mianowicie, że mając to samo na wejściu, produkują to samo na wyjściu. Mówiąc, że X jest modelem procesu A, mamy jednak na myśli coś więcej – model powinien też być do modelowanego procesu podobny strukturalnie. Oznacza to, że korelacje powinny zachodzić nie tylko pomiędzy wejściami i wyjściami procesów X i A, ale i poszczególnymi etapami transformacji, które w obu procesach zachodziły [Cummins, 1989]. Mówiąc o podobieństwie (a nie identyczności) struktur transformacji, uzyskujemy też dość poręczne narzędzie – możemy bowiem mówić o bardziej i mniej doskonałych modelach, symulację uznając za graniczny przykład modelu, w którym podobne zostały już tylko pary wejść i wyjść.

Sądzę, że wygodnym z punktu widzenia filozofii informatyki rozwiązaniem jest uznanie programów wykonywanych przez komputery za S-reprezentacje tych procesów, których są one modelami. Wymagane do bycia S-reprezentacją podobieństwo strukturalne zachodzi w tym przypadku pomiędzy dynamiczną strukturą procesu a strukturą programu. Na przykład – wspomniany program do obliczania liczby dni, które minęły od danej daty, jest S-reprezentacją tego okresu, ponieważ struktura mierzonej wielkości jest podobna (np. większe jednostki dzielą się na tę samą liczbę mniejszych) oraz proporcje pomiędzy wielkościami, którymi manipuluje program, odzwierciedlają proporcje zachodzące pomiędzy obiema branymi pod uwagę datami. Raz jeszcze podkreślmy, że relacja ta nie implikuje odniesienia – jeżeli jakiś program równie dobrze odzwierciedla dwie różne struktury, to jest równie dobrą S-reprezentacją obu.

Posłużenie się pojęciem S-reprezentacji pozwala nam zdać sprawę z interesujących różnic i zależności pomiędzy programami a ich fizycznymi realizacjami, czyli uwypukla dokładnie ten problem, który reprezenta-

cje symboliczne zacierają. Możemy na przykład oddzielnie pytać o to, czy dany program/implementacja bliższy jest modelowi czy symulacji, albo poszukiwać korelacji pomiędzy strukturą programu a możliwymi architekturami, które go realizują. Dla przykładu – program napisany w pseudokodzie może stanowić bardzo szczegółowy model zjawiska, ale jego implementacja w konkretnym języku programowania może część podobieństwa strukturalnego zatracić (na przykład ze względu na specyfikę składni tego języka albo na jakieś arbitralne ograniczenia dotyczące typów stosowanych w nim zmiennych). Programista może na przykład wykorzystać tę samą zmienną dwa razy do reprezentowania dwóch różnych wartości (ponieważ wie, że struktura programu uniemożliwia powstanie konfliktu)[11]. Możemy sobie następnie wyobrazić, że fizyczna realizacja programu zmusza do dalszych kompromisów. Skompilowany program dostosowany do fizycznej architektury komputera może w jeszcze większym stopniu przesunąć się w stronę symulacji (odchodząc tym samym od modelu). Ku naszemu zdziwieniu może się wtedy na przykład okazać, że fizyczna realizacja programu będącego modelem zjawiska A sama w sobie jest raczej symulacją zjawiska A, ale równolegle niezłym modelem jakiegoś innego zjawiska B. Hipotezy i rozróżnienia, które dadzą się dzięki pojęciu S-reprezentacji formułować, można mnożyć. Obserwując działanie emulatora, możemy na przykład stwierdzić, że jeden z jego modułów jest S-reprezentacją jakiegoś podzespołu emulowanej maszyny, a inny jedynie ją symuluje. W skrócie – pojęcie S-reprezentacji dostarcza jednolitej miary do oceny dwóch bardzo różnych zjawisk (algorytmu i fizycznej implementacji), nie wymagając przy tym ani relacji odniesienia, ani zewnętrznego obserwatora.

Pojęcie S-reprezentacji (rozumianej jako całościowe podobieństwo strukturalne) pozwala nam też na wprowadzenie dodatkowego podtypu reprezentacji, którego Peirce nie antycypował. Aby zrozumieć jego specyfikę, wróćmy jeszcze na moment do naszego przykładu z mapą. Jak wspomnieliśmy, mapa reprezentuje pewien obszar, ponieważ jest jego S-reprezentacją. Zauważmy jednak, że w przypadku wielu części mapy tak

[11] Praktyki tego rodzaju są zazwyczaj potępiane, ale w pewnych specyficznych warunkach mogą stać się konieczne (np. w sytuacji, w której każda inicjalizacja nowej zmiennej wpływa na bardzo ograniczone zasoby pamięci).

nie jest. Oznaczone za pomocą kropki miasto nie jest S-reprezentacją miasta. Mimo to mówimy, że „reprezentuje ono miasto". Jakiego rodzaju jest to reprezentacja? Czy jest to reprezentacja symboliczna? Wydaje się, że nie – reprezentacją symboliczną jest nazwa miasta znajdująca się obok kropki. Na dodatek pozostałe miasta oznaczone są takimi samymi kropkami. Czy jest to reprezentacja wskaźnikowa? Nic na to nie wskazuje – kropka ta nie została przez to miasto na mapie wytworzona i nie zniknie, gdy miasto przestanie istnieć, nie jest więc z miastem powiązana przyczynowo. Reprezentację tego typu najwygodniej będzie nam nazwać S-reprezentacją wtórną. Kropka reprezentuje bowiem miasto tylko dlatego, że: (1) jest częścią struktury, która jako całość jest S-reprezentacją terenu, którego to miasto jest częścią i (2) odgrywa w ramach tej struktury rolę, która jest strukturalnie podobna do roli, którą odgrywa miasto w terenie (na przykład jest w środku mapy, dokładnie tak jak miasto, które jest w środku terenu).

Sądzę, że pojęcie S-reprezentacji wtórnej okazuje się szczególnie przydatne w kontekście fragmentów programów – zmiennych, procedur, funkcji czy nawet całych bloków. Potoczny sposób mówienia o elementach programów jest bowiem dość nieporządny i natychmiast przywodzi na myśl odniesienie przedmiotowe. Jak wspomnieliśmy już wcześniej, o zmiennych mówimy często, że jakąś wielkość czy strukturę „reprezentują" lub, że się do niej „odnoszą". Warto zauważyć, że nie jest to specyfika języka polskiego. Angielski idiom „to stand for", używany zamiennie z „to represent", również natychmiast przywodzi na myśl relację odniesienia przedmiotowego i równie powszechnie używany jest w kontekście fragmentów kodu. Jak widzieliśmy powyżej, niekiedy oznacza to tylko tyle, że użytkownik wykorzystuje tę zmienną w określony sposób. Niekiedy sens tego potocznego sformułowania może być jednak inny i sprowadza się do tego, że zmienna jest S-reprezentacją wtórną danej wielkości. Pamiętajmy, że nie implikuje to wcale odniesienia przedmiotowego. Zmiennej w programie, zupełnie tak jak kropki na mapie, nie łączy z oznaczaną przez nią wielkością żadna tajemnicza relacja. Niewykluczone, że zmienna ta oznacza także inne wielkości. Wystarczy, że tak się złoży, że istnieje gdzieś w świecie inna struktura podobna do całości programu, a jakiś element

odgrywa w niej podobną rolę[12]. Choć mówiliśmy dotąd przede wszystkim o algorytmach, to warto podkreślić, że niemałą zaletą S-reprezentacji jest łatwość, z jaką stosują się do struktur danych. Dla przykładu – wielowymiarowa tablica zawierająca dane opisujące jakiś fragment rzeczywistości (na przykład proces produkcyjny w jakimś przedsiębiorstwie) przejawia strukturalne podobieństwo do tego fragmentu rzeczywistości. Nie trzeba chyba dodawać, że jeszcze lepiej widoczne jest to w przypadku takich struktur danych, jak drzewa.

Nie oznacza to, rzecz jasna, że z pojęciem S-reprezentacji nie wiążą się pewne kontrowersje. Wręcz przeciwnie – wzbudza ono na gruncie kognitywistyki wiele sporów. Są to jednakże kontrowersje dość ogólnej natury, które nie są bezpośrednio powiązane z pytaniem o ich przydatność dla filozofii informatyki[13]. Nie mam tu miejsca na szczegółową analizę i obronę zarzutów, które wobec S-reprezentacji wysunięto. Wspomnę jedynie o dwóch, które (jeśli okażą się zasadne) mogą okazać się dla nas istotne.

Po pierwsze, wydaje się, że homomorfizm jest dość liberalną relacją i jeśli ktoś bardzo się uprze, to może dowodzić, że wszystko jest S-reprezentacją wszystkiego. W interesującym nas kontekście oznaczałoby to, że o każdym programie można by powiedzieć, że modeluje każde zjawisko. Argumenty przeciw tej tezie wysuwałem już w innym miejscu [Grabarczyk, 2013], wspomnę więc jedynie, że tego rodzaju argumenty oparte są zawsze na wprowadzeniu bardzo zręcznego zewnętrznego interpretatora, który, jak na zawołanie, odpowiednie podobieństwo strukturalne znajduje. Bycie S-reprezentacją powinno się jednakże dać wykryć także bez pomocy owego sprawnego interpretatora.

Po drugie, niektórzy badacze twierdzą, że rozróżnienie pomiędzy S-reprezentacjami i reprezentacjami wskaźnikowymi nie daje się przepro-

[12] Zauważmy, że mowa tu o podobieństwie (rozumianym jako możliwość dokładniejszego albo mniej dokładnego odwzorowania), a nie identyczności. Dlatego też nie można wykluczyć przypadku, w którym dość podobna struktura zawiera elementy, których rozłożenie w tej strukturze się różni. Przynależność do S-reprezentacji nie gwarantuje zatem bycia S-reprezentacją wtórną.

[13] Pomijam tu, rzecz jasna, przypadek ewentualnej kompletnej dyskredytacji tego pojęcia (np. wykazania, że jest wewnętrznie sprzeczne).

wadzić, ponieważ wszystkie wskaźniki są częściowymi S-reprezentacjami [Morgan, 2014]. Argumentem tym zajmuję się szczegółowo w innym miejscu [Grabarczyk, 2015]. Warto jednakże zauważyć, że w przypadku przedstawianego przeze mnie obrazu oznacza to jedynie konieczność zgrupowania drugiego i trzeciego typu reprezentacji pod jednym nagłówkiem. Jak starałem się pokazać, reprezentacje wskaźnikowe również mają swoje miejsce w filozofii informatyki i nie prowadzą do żadnych nieprzezwyciężonych trudności. Kłopoty mielibyśmy jedynie wtedy, gdyby coś zmusiło nas do przyjęcia pojęcia reprezentacji symbolicznej, a to, jak się wydaje, w ogóle nam nie grozi.

A zatem, posługując się na gruncie filozofii informatyki terminem „reprezentacja", musimy wyraźnie wskazywać na to, o który typ reprezentacji chodzi. Z trzech (wyróżnionych za Peirce'em) znaczeń terminu „reprezentacja" jedynie dwa okazały się rzeczywiście użyteczne dla dziedziny. Są to: reprezentacje wskaźnikowe, które pozwalają na wygodny opis manipulacji danymi pochodzącymi z sensorów maszyny, oraz S-reprezentacje, które pozwalają na opis programu wziętego w całości oraz na dość precyzyjne rozróżnianie między modelami i symulacjami.

Reprezentacje symboliczne, czyli paradoksalnie te, o których mowa najczęściej, okazują się zaś zupełnie zbędne. Odwoływanie się do nich bazuje bowiem głównie na nieporozumieniach – apriorycznym założeniu o referencyjnym charakterze symboli, mieszaniu relacji synonimiczności z odniesieniem przedmiotowym i przywiązaniu do potocznego sposobu mówienia. Co gorsza, okazują się również szkodliwe, ponieważ powoływanie się na reprezentacje symboliczne wymusza na nas wprowadzenie filozofii informatyki pojęć „odniesienia przedmiotowego" i „interpretatora", bez których doskonale mogłaby się ona obyć. To, w gruncie rzeczy, dobra wiadomość. Okazuje się bowiem, że w tych znaczeniach, w których pojęcie reprezentacji może się filozofom informatyki przydać, jest ono wolne od paradoksów i kłopotliwych konsekwencji. W tych zaś znaczeniach, w których pojęcie to prowadzi do trudnych do przezwyciężenia trudności, jest ono jednocześnie eksplanacyjnie jałowe i może zostać z powodzeniem zastąpione bezpieczną parafrazą.

Bibliografia

Borges J.L., (1998), „O ścisłości w nauce", [w:] J.L. Borges, *Twórca*, Warszawa, Prószyński i S-ka.

Brooks R.A., (1990), "Elephants don't play chess", *Robotics and Autonomous Systems* 6, s. 3–15.

Brooks R.A., (1991), "Intelligence without representation", *Artificial Intelligence* 47, s. 139–159.

Campos D., (2009), "Imagination, concentration, and generalization: Peirce on the reasoning abilities of the mathematician", *Transactions of the Charles S. Peirce Society* 45(2), s. 135–156.

Chalmers D., (1996), "Does a rock implement every finite-state automaton?", *Synthese* 108, s. 309–333.

Chrisley L.R., (1994), "Why everything doesn't realize every computation?", *Minds and Machines*, 4, s. 403–420.

Copeland J., (1996), "What is computation?", *Synthese* 108, s. 335–359.

Crane T., (2003), *The Mechanical Mind: A Philosophical Introduction to Minds, Machines, and Mental Representation*, New York, Routledge.

Cummins R., (1989), *Meaning and Mental Representation*, Cambridge, MA, MIT Press.

Dietrich E., Markman A.B., (2003), "Discrete thoughts: Why cognition must use discrete representations", *Mind and Language* 18, s. 95–119.

Dretske F., (1988), *Explaining Behavior*, Cambridge, MA, MIT Press.

Dretske F., (1995), *Naturalizing the Mind*, Cambridge, MA, MIT Press.

Field H., (1978), "Mental representation", *Erkenntnis* 13, s. 9–61.

Fodor J.A., (1981), "The mind-body problem", *Scientific American*, 244, s. 114–123.

Fodor J.A., (1998), *Concepts: Where Cognitive Science Went Wrong*, Oxford, Oxford University Press.

Fresco N., (2010), "Explaining computation without semantics: keeping it simple", *Minds & Machines*, 20, s. 165–181.

Grabarczyk P., (2013), "O niearbitralnym kryterium posiadania struktury obliczeniowej", *Filozofia Nauki*, s. 31–50

Grabarczyk P., (2015), "Concepts as soft detectors – On the role concepts play in perception", *New Ideas in Psychology*, Vol. 40, Part A, s. 86–93

Harnad S., (1990), "The symbol grounding problem", *Physica* D 42, s. 335–346.

Hutto D., Myin E., (2013), *Radicalizing Enactivism: Basic Minds without Content*, Cambridge, MA, MIT Press.

Leibniz G.W., (1991), *Monadologia*, Toruń, Wydawnictwo UMK.

Millikan R., (1984), *Language, Thought and Other Biological Categories*. Cambridge, MA, MIT Press.

Millikan R., (2004), *Varietes of Meaning*. Cambridge, MA, MIT Press.

Miłkowski M., (2013), *Explaining the Computational Mind*, Cambridge, MA, MIT Press.

Morgan A., (2014), "Representations gone mental", *Synthese* 191.2, s. 213–244.
Morris M., (1991), "Why there are no mental representations", *Minds and Machines*, Vol. 1, nr 1, s. 1–30.
Müller V.C., (2007), "Is there a future for AI without representation?", *Minds and Machines* 17.1, s. 101–115.
Müller V.C., (2008), "Representation in digital systems", [w:] A. Briggle i in. [eds.], *Current Issues in Computing and Philosophy*, Amsterdam, IOS Press.
Newell A., (1980), "Physical symbol systems", *Cognitive Science* 4.2, s. 135–183.
O'Brien G., (1988), "Connectionism, analogicity and mental content", *Acta Analytica* 22, s. 111–131.
Piccinini G., (2008), "Computation without representation", *Philosophical Studies*, 137, s. 205-241
Pylyshyn Z.W., (1984), *Computation and Cognition*, Cambridge, MA, The MIT Press.
Ramsey W.M., (2007), *Representations Reconsidered*, Cambridge, MA, Cambridge University Press.
Searle R.J., (1980), "Minds, brains and programs", *Behavioral and Brain Sciences*, Vol 3, nr 3, s. 417–457.
Searle R.J., (1990), „Is the brain a digital computer?", *Proceedings and Addresses of the American Philosophical Association*, Vol. 64, No. 3. s. 21–37.
Shagrir O., (2010), "Towards a modelling view of computing", [w:] G. Dodig-Crnkovic, M. Burgin [eds.], *Information and Computation*, Singapore, World Scientific Publishing.
Shannon C.E., (1948), "A mathematical theory of communication", *Bell System Technical Journal* 27 (3), s. 379–423.
Shepard R.N., Chipman S. (1970), "Second-order isomorphism of internal representations: shapes of states", *Cognitive Psychology* 1, s. 1–17.
Smith B.C., (1995), *The Origin of Objects*, Cambridge, MA, MIT Press.
Smith B.C., (2002), "The foundations of computing", [w:] M. Scheutz [ed.], *Computationalism: New Directions*, s. 23–58 Cambridge, MA, MIT Press.
Steiner P., (2013), "C.S. Peirce and artificial intelligence: historical heritage and (new) theoretical stakes", [w:] V. Müller [ed.], *Studies in Applied Philosophy, Epistemology and Rational Ethics. Philosophy and Theory of Artificial Intelligence*, Springer, s. 265–276.
Swoyer C. (1991), "Structural representation and surrogative reasoning", *Synthese*, 87, s 449–508.
Turing A., (1950), "Computing machinery and intelligence", *Mind* 59, s. 433–460.

Three types of representations in the philosophy of computer science

ABSTRACT. This paper is devoted to the role that the notion of representation plays in the philosophy of computer science. The analysis that is presented here uses the classic

division of symbolic representations, indices and icons, as proposed by C.S. Peirce. The author of this paper argues that, although it is referred to in the literature most often, the first type of representations can be safely eliminated from the discipline. However, the second and third types have important applications in the philosophy of computer science.

KEY WORDS: representations, philosophy of computer science, s-representations, symbolic representations, indices

Paweł Grabarczyk, Instytut Filozofii Uniwersytetu Łódzkiego, ul. Kopcińskiego 16/18, 90-232 Łódź, pagrab@gmail.com.