Logika jako narzędzie w technice informatycznej

Piotr Kulicki

1. Wstęp

Systemy sterowane przez komputer, bądź korzystające z komputerów w inny istotny dla nich sposób są w dzisiejszym świecie omalże powszechne. Wymienić tu można na przykład automatycznego pilota w samolocie, sterowane komputerowo linie produkcyjne, systemy ogrzewania i wentylacji w dużych budynkach, ale również komputerowe systemy ewidencji ludności czy też rachunkowości przedsiębiorstwa. Jak w każdym innym wytworze człowieka w systemie komputerowym mogą pojawić się błędy. Są one zazwyczaj spowodowane przez wadliwe działanie programów, które są realizowane w tym systemie. Nawet drobny błąd w programie, ze względu na automatyczne jego wykonanie, może spowodować poważne konsekwencje. Nietrudno sobie wyobrazić, że konsekwencje te mogą być tragiczne dla użytkowników, ale system komputerowy może być również po prostu niewygodny, i zamiast ułatwiać życie swoim adresatom, powodować kłopoty.

W tej sytuacji jest oczywiste, że wszystko co może pozwolić na wyeliminowanie błędów jest bardzo wartościowe. Jak przy innych „sztukach" w programowaniu komputerów można zauważyć dwa typowe podejścia, nazwijmy je umownie „rzemieślniczym" i „inżynierskim" (w praktyce są one zazwyczaj mieszane). Pierwsze z nich opiera się na doświadczeniu osoby wykonującej rzemiosło i jej poprzedników. Wykonujemy czynności w pewien sposób, ponieważ tak zostaliśmy nauczeni przez mistrza i przynosiło to zazwyczaj pozytywny rezultat. W drugim podejściu opieramy się na badaniu naukowym. Pozwala ono na lepsze zrozumienie prowadzonej działalności i racjonalnie uzasadnioną pewność, że coś jest zrobione dobrze.

Chociaż wprawny „rzemieślnik" może działać sprawniej od „inżyniera", ten drugi działa pewniej. Przesunięcie od rzemiosła w stronę inżynierii także w przypadku sztuki programowania komputerów sprawia, że mamy większe możliwości eliminowania błędów. Jest tak ponieważ możemy wyraźniej sformułować zasady, którymi powinniśmy się kierować, a także dlatego, że same te zasady są pewniejsze.

W niniejszej pracy wskażemy na pewne elementy związane z przejściem od rzemiosła do nauki w programowaniu komputerów, szczególną uwagę zwracając na udział w nim logiki. W kolejnych paragrafach, w sposób bardzo syntetyczny ze względu na rozmiary pracy, przeanalizujemy proces powstawania programu komputerowego, sposoby opisu działania

komputera oraz sposoby wykazywania poprawności programów. W podsumowaniu zbierzemy uwagi dotyczące wykorzystania logiki dla tworzenia poprawnie działających programów.

Podstawowe zagadnienia związane z programowaniem komputerów i ich funkcjonowaniem przedstawione są tu zgodnie z panującymi współcześnie w informatyce teoretycznej poglądami (por. np. [2]). Nowością ujęcia przedstawionego w pracy jest wyraźne uznanie roli logiki jako nauki podstawowej w stosunku do informatyki i zebranie miejsc, w których odgrywa ona istotną rolę.

2. Tworzenia programu komputerowego

Proces tworzenia programu komputerowego jest złożony i niejednorodny. Pierwszym jego etapem jest rozpoznanie i analiza problemu, który ma być rozwiązywany przez program. Aby program mógł zostać wdrożony analiza ta musi uwzględniać możliwie szeroki kontekst przyszłego działania systemu, obejmować rozważenie możliwych korzyści z komputeryzacji oraz wszelkiego rodzaju zagrożeń. System komputerowy zazwyczaj zastąpić ma jakąś działalność ludzką odbywającą się bez komputera. Twórca programu musi więc zapoznać się z tą działalnością i zadecydować, co z niej ma pozostać, a co może być usprawnione bądź pominięte przy użyciu komputerów.

Wyniki tej analizy muszą następnie być w miarę możliwości precyzyjnie zwerbalizowane w postaci specyfikacji, najlepiej formalnej, oczekiwanego działania programu. Specyfikacja ta powinna być zrozumiała dla użytkownika i zdobyć jego akceptację, pozostając podstawą dla dalszej pracy. Następnie programista znaleźć musi techniczny sposób zrealizowania tak określonego programu. Zazwyczaj podstawowym zadaniem jest tu znalezienie optymalnego algorytmu postępowania gwarantującego realizację postawionych celów. Ostatnim etapem opisywanego procesu jest wyrażenie rezultatów poprzednich etapów w postaci „zrozumiałej” dla komputera, tzn. w postaci instrukcji w języku programowania.

Upraszczając nieco można powiedzieć, że w historycznym rozwoju informatyki uwaga przenosi się z końca na początek tego procesu. Początkowo najwięcej uwagi poświęcano napisaniu programu. Sposób komunikowania się z komputerem był skomplikowany, trzeba było rozpisywać algorytmy na podstawowe instrukcje i pochłaniało to dużo pracy. Użycie języków wysokiego poziomu, w których definiować można procedury i łączyć osobno opracowywane moduły w większe całości pozwala na łatwiejsze pisanie programów i zajęcie się pozostałymi etapami pracy. Równolegle, aby program mógł być wykonany w rozsądnym

czasie trzeba było pracować nad szybkimi algorytmami rozwiązania problemu. Problem optymalizacji algorytmów również traci swoje pierwszorzędne znaczenie. Jest tak z dwóch powodów: po pierwsze ze względu na to, że optymalne algorytmy dla większości podstawowych problemów powtarzających się w programach zostały już opracowane i opisane, po drugie ze względu na ciągle rosnące możliwości sprzętu komputerowego.

Zauważmy teraz, że z punktu widzenia użytkownika to nie program, ani nawet nie algorytm są interesujące. Ważne jest natomiast, aby pożądany sposób działania programu był dobrze określony, oraz żeby program go realizował w pewny i niezawodny sposób. Wysiłek poświęcony budowaniu programu, a zarazem jego sukces tkwi coraz bardziej w pierwszym etapie jego budowy. Etap ten jest również najbardziej twórczy. Należy wymyślić jak program ma działać, a samo napisanie programu jest już zazwyczaj powielaniem znanych wzorców, które zapewnią, że powstający w rezultacie program realizuje specyfikację.

3. Opis działania komputera

Aby udowodnić, że komputer będzie realizował nasze zamierzenia musimy mieć możliwość ścisłego opisu jego funkcjonowania w interesującej nas sytuacji. Opis taki może być dokonany na różnych poziomach abstrakcji, od poziomu fizycznych urządzeń do realizacji poszczególnych fragmentów programu komputerowego napisanego w języku wysokiego poziomu, bądź nawet poszczególnych fragmentów specyfikacji.

Takie wielopoziomowe ujęcie jest w nauce często spotykane. W analogiczny sposób rozpatrywane są zjawiska fizyczne (poziom cząstek elementarnych, atomów, molekuł, zjawisk obserwowalnych przez nas bezpośrednio), biologiczne (zjawiska na poziomie molekularnym, komórek, tkanek, organów, organizmów, gatunków, ekosystemów) psychiczno-społeczne (uwarunkowania biologiczne zachowania człowieka, działanie świadome jednostki, funkcjonowania małych grup społecznych i całych społeczeństw). Różnice pomiędzy poziomami mogą polegać na skali zjawisk bądź na aspekcie naszego ich postrzegania.

W przypadku opisu funkcjonowania komputera możemy wyróżnić następujące zasadnicze poziomy abstrakcji[1]:

1. proste urządzenia fizyczne

[1] Tradycyjnie w informatyce rozróżnia się poziom sprzętowy i poziom oprogramowania. Różnica ta jest jednak płynna i nie jest kluczowa z naszego punktu widzenia - nie jest ważne, czy coś jest realizowane sprzętowo czy też programowo. Wskazuje to na istotną spójność informatyki, a w szczególności tych dwóch często wyraźnie odróżnianych jej działów

2. obwody kombinatoryczne
3. obwody sekwencyjne z pamięcią
4. programy w kodzie maszynowym
5. programy w językach wysokiego poziomu
6. specyfikacja

Scharakteryzujemy krótko każdy z wymienionych poziomów, zwracając szczególną uwagę na jego związek z logiką.

Ad 1. Na tym poziomie abstrakcji znajdujemy się właściwie na terenie elektroniki, a nie informatyki. Tak jak w większości nauk technicznych prowadzone prace zmierzają do osiągnięcia, w oparciu o rezultaty nauki podstawowej (fizyki, chemii, biologii), jakiegoś efektu w świecie materialnym, w tym wypadku zgodnego z naszymi oczekiwaniami przepływu prądu przez urządzenie.

W dalszym ciągu naszych rozważań przejdziemy na grunt specyficzny dla informatyki. Nie będzie nas już interesował efekt w świecie materialnym, ale jego interpretacja, tzn. niesiona przez niego informacja. Prąd w obwodzie będzie traktowany jako znak odnoszący nas do czegoś innego, a nie jako zjawisko fizyczne. Oczywiście aby tak mogło być fizyczny aspekt urządzenia musi być w interesującym nas zakresie przewidywalny.

Ad 2. Mając do dyspozycji dobrze skonstruowane urządzenie elektroniczne możemy przejść do jego interpretacji informatycznej. Okazuje się, że do opisu na tym poziomie doskonale nadaje się logika, a konkretnie klasyczny rachunek zdań. Fizyczne stany przewodów interpretujemy analogicznie do wartości logicznych - wysoki potencjał jako 1 (prawda), niski - 0 (fałsz). Złożone struktury fizyczne traktujemy jak formuły logiczne. Taka interpretacja pozwala na dowolne przekształcenie danych wejściowych na dane wyjściowych, o ile ujęte są one w postaci kombinacji zer i jedynek. Klasyczny rachunek zdań jest tu narzędziem do projektowania obwodów.

Ad 3. Na poziomie obwodów kombinatorycznych ujmujemy stan urządzenia elektronicznego w konkretnym momencie czasowym. Dla działania komputera bardzo istotne są natomiast zmiany zachodzące w czasie. Ujęcie takich zmian możliwe jest obwodach sekwencyjnych, które powstają przez dodanie do obwodów kombinatorycznych aspektu czasowego. Zaznacza się moment zaistnienia stanu urządzenia, a także zasady zmiany jaka dokonuje się przy przejściu do następnego stanu. Użycie pamięci zwiększa możliwości obwodu sekwencyjnego i umożliwia modyfikacje jego działania poprzez dostarczanie do

pamięci różnych zestawów danych. Na tym poziomie można tak samo jak na poprzednim skorzystać z osiągnięć logiki używając do opisu zmian zachodzących w czasie logiki temoporalnej.

Ad 4. Wykorzystanie obwodu sekwencyjnego z pamięcią można w dużym stopniu modyfikować poprzez zmianę parametrów zapisanych w pamięci. Parametry te mogą przybierać postać programu, który zawiera ciąg poleceń, które z kolei określają zmianę zachowania się obwodu i stanu pamięci. Dzięki temu obwód sekwencyjny staje się wielofunkcyjny.

Ad 5. O ile program w języku maszynowym zawiera instrukcje dotyczące prostych czynności do wykonania w obwodzie sekwencyjnym, program w języku wysokiego poziomu zawiera polecenia składające się na algorytm rozwiązania zadania. Instrukcje programu nie są więc związane bezpośrednio z maszyną na jakiej mają zostać wykonane, mogą więc być realizowane przy użyciu różnych urządzeń.

Ad 6. Specyfikacja jest ujętym w możliwie jak najbardziej formalnym języku opisem pożądanego działania programu. Ze względu na to, że specyfikacja powinna być z jednej strony możliwie ścisła, a z drugiej strony może dotyczyć bardzo różnorodnych problemów (nie tylko natury liczbowej) do jej pisania bardzo dobrze nadaje się język logiki, a w szczególności rachunku predykatów, rachunku relacji oraz różnych wariantów logiki temporalnej (por. np. [1], [2], [4]).

4. Poprawność programu

Aby udowodnić, że specyfikacja jest faktycznie realizowana musimy przejść od poziomu najwyższego do najniższego pokazując, że spełnienie wymogów w stosunku do zjawisk obserwowanych na poziomie wyższym jest zagwarantowane przez prawidłowości dotyczące poziomu niższego.

Przejście od poziomu programu do fizycznej realizacji dokonuje się poza programistą i nie będziemy się nim szczegółowo zajmować. Poprawność przejścia pomiędzy nimi jest normalnie wykazywana metodami naukowymi. Poszczególne etapy zagwarantowane są odpowiednio przez prawidłową konstrukcję translatora języka programowania, budowę fizycznej bądź wirtualnej maszyny abstrakcyjnej, ujętej jako obwód sekwencyjny z pamięcią, który z kolei zawiera prawidłowo ze sobą połączone obwody kombinatoryczne zrealizowane przez układ tranzystorów i przewodów je łączących.

Poprawność przejścia od poziomu specyfikacji do programu w języku wysokiego poziomu pozostaje w gestii programisty i często pozostawiana jest jego wyczuciu (podejście „rzemieślnicze"). Tak jednak być nie musi. W przypadku imperatywnego paradygmatu programowania[2] można je formalnie wykazać korzystając z denotacyjnej semantyki języka programowania (zob. np. [2]).

W semantyce denotacyjnej ujmuje się znaczenie programu jako relację pomiędzy stanem początkowym a końcowym wykonania tego programu. Określa się w jaki sposób dane wyjściowe otrzymane po wykonaniu programu wyznaczone są przez dane wejściowe i program. Semantykę taką buduje się poprzez określenie znaczenia elementarnej czynności jaką jest przypisanie wartości zmiennej oraz najprostszych operacji składania programów bardziej skomplikowanych z prostszych elementów, którymi są warunkowe wykonanie programu i złożenie sekwencyjne oraz złożenie równoległe programów. Wszelkie pozostałe konstrukcje występujące w programach mogą być sprowadzone do kombinacji tych elementarnych.

W semantyce denotacyjnej znaczenie programu jest zatem wyrażone w języku rachunku predykatów i może być bezpośrednio porównywane z wyrażoną w tym samym języku specyfikacją. Pozwala to na pokazanie na gruncie logiki (rachunku predykatów) formalnego dowodu poprawności programu.

Pójściem dalej w tym samym kierunku jest użycie języków deklaratywnych, np. Prologu (skrót od „*Pro*gramowanie w *log*ice"). W Prologu sam program jest napisany w języku rachunku predykatów (zob. np. [3]), a więc w tym samym, w którym normalnie pisze się specyfikację. W rezultacie mamy specyfikację wyrażoną w nieco ściślej określonej postaci formalnej, która jednocześnie jest wykonywalnym programem. Unika się przekładania, które może powodować pomyłki. Zamiast dowodu, że specyfikacja i program pokrywają się dokonujemy automatycznego przekształcenia jednego w drugie. Jest to wygodniejszy i pewniejszy sposób postępowania.

5. Podsumowanie

Dla bezpiecznego zastosowania systemów komputerowych istotne jest przejście w programowaniu od czysto praktycznej umiejętności tworzenia programów, do programowania

[2] Języki programowania wysokiego poziomu można podzielić na dwie grupy: imperatywne i deklaratywne. W językach imperatywnych program składa się z serii poleceń dotyczących wartości zmiennych rozumianych jako komórki pamięci. Języki deklaratywne natomiast zawierają definicje funkcji lub relacji, które pozwalają na obliczenie ich wartości dla interesujących nas parametrów.

traktowanego jako dyscypliny naukowej. W tym przejściu dużą rolę odgrywa zastosowanie w informatyce logiki. Ujawnia się to najwidoczniej w następujących momentach:

- Język logiki (głównie rachunku predykatów, rachunku relacji i logik temporalnych) dobrze nadaje się do pisania specyfikacji.
- Teorie logiczne (klasyczny rachunek zdań, logiki temporalne) są narzędziem dla dowodów poprawności realizacji programu przez komputer.
- Użycie logiki pozwala na udowodnienie poprawności programu względem specyfikacji.
- Specyfikacja w języku logiki może stać się wykonywalnym programem (Prolog).

Interesujące i pouczające jest to, że logika powstała wiele wieków temu niezależnie od jakichkolwiek rozwiązań technicznych znajduje w informatyce techniczne zastosowanie. Prowadzi to do wzajemnego oddziaływania logiki i informatyki - logika jest przydatna w informatyce, ale również rozwija się poprzez znalezienie nowych problemów.

Użycie logiki wpływa na racjonalizacje budowania systemów komputerowych, stają się one bardziej zrozumiałe i w ten sposób bardziej ludzkie. Nie powstrzyma to nas jednak przed nieracjonalnymi pomysłami stosującymi pewne technologie, tylko dlatego, że jest to możliwe, a nie dlatego, że wiadomo iż ich wykorzystanie będzie pożyteczne. Może natomiast sprawić, że lepiej będziemy mogli zrozumieć, co chcemy osiągnąć.

Bibliografia

[1]Hajnicz, E., Reprezentacja logiczna wiedzy zmieniającej się w czasie, Warszawa 1996.

[2] Hoare, C.A.R., Mathematic Models for Computing Science, Lecture Notes for Marktoberdorf Summer School, 1994.

[3] Kowalski, R., Logic for problem-solving, Edinburgh 1974.

[4]von Karger, B., Hoare, C.A.R., Sequential calculus, Information Processing Letters 53 (1995), str. 123-130.