# KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Synteza programów [S2SI1E>SPR]

## Przedmiot

Kierunek studiów
Sztuczna inteligencja/Artificial Intelligence

Rok/Semestr
1/1

Studia w zakresie (specjalność)
–

Profil studiów
ogólnoakademicki

Poziom studiów
drugiego stopnia

Język oferowanego przedmiotu
angielski

Forma studiów
stacjonarne

Wymagalność
obligatoryjny

## Liczba godzin

Wykład
30

Laboratorium
0

Inne (np. online)
0

Ćwiczenia
0

Projekty/seminaria
30

## Liczba punktów ECTS

4,00

## Koordynatorzy

dr inż. Iwo Błądek
iwo.bladek@put.poznan.pl

prof. dr hab. inż. Krzysztof Krawiec
krzysztof.krawiec@put.poznan.pl

## Wykładowcy

## Wymagania wstępne

Student rozpoczynający ten kurs powinien posiadać podstawową wiedzę z zakresu logiki, algorytmów optymalizacji kombinatorycznej oraz uczenia maszynowego. Powinien również posiadać umiejętność pozyskiwania informacji ze wskazanych źródeł oraz współpracy w zespole, gdyż kurs zakłada realizację projektów grupowych.

## Cel przedmiotu

1. Zapoznanie studentów z podstawami i wybranymi zaawansowanymi zagadnieniami syntezy programów, z naciskiem na powiązania ze sztuczną inteligencją i uczeniem maszynowym. 2. Rozwijanie umiejętności rozwiązywania problemów związanych z syntezą programów, w szczególności formułowania specyfikacji własności, jakie powinien mieć syntetyzowany program, wyboru odpowiedniej techniki syntezy programów oraz oceny jej wyników. 3. Rozwijanie umiejętności pracy zespołowej nad projektem programistycznym poprzez łączenie studentów w pary do realizacji projektu podczas laboratoriów.

## Przedmiotowe efekty uczenia się

Wiedza
Ma uporządkowaną i podbudowaną teoretycznie wiedzę ogólną dotyczącą kluczowych zagadnień z zakresu syntezy programów [K2st_W2].
Ma zaawansowaną wiedzę szczegółową dotyczącą wybranych zagadnień z zakresu syntezy programów, w szczególności różnych typów specyfikacji programów, stosowania logicznego rozumowania w celu zapewnienia poprawności programu i wnioskowania o programie spełniającym specyfikację, oraz rozumienia, w jaki sposób techniki optymalizacji stochastycznej i uczenia maszynowego mogą być wykorzystane do rozwiązywania lub ułatwiania rozwiązywania problemów syntezy programów [K2st_W3].
Ma zaawansowaną i szczegółową wiedzę na temat procesów zachodzących w cyklu życia systemów syntezy programów, w tym technik pozyskiwania danych oraz projektowania, testowania i wdrażania takich systemów [K2st_W5].
Zna zaawansowane metody, techniki i narzędzia stosowane do rozwiązywania złożonych zadań inżynierskich i prowadzenia badań w zakresie syntezy programów, w szczególności metodologię dotyczącą prowadzenia eksperymentów obliczeniowych oraz metryki oceny jakości systemów syntezy programów [K2st_W6].

Umiejętności
Potrafi planować i przeprowadzać eksperymenty, w tym pomiary i symulacje komputerowe, interpretować uzyskane wyniki i wyciągać wnioski oraz formułować i weryfikować hipotezy dotyczące złożonych problemów inżynierskich oraz prostych problemów badawczych w zakresie syntezy programów [K2st_U3].
Potrafi wykorzystać metody analityczne, symulacyjne i eksperymentalne do formułowania i rozwiązywania problemów inżynierskich oraz prostych problemów badawczych w zakresie syntezy programów [K2st_U4].
Potrafi - przy formułowaniu i rozwiązywaniu zadań inżynierskich - integrować wiedzę z różnych obszarów informatyki (a w razie potrzeby także wiedzę z innych dyscyplin naukowych) oraz stosować podejście systemowe, uwzględniając także aspekty pozatechniczne [K2st_U5].
Potrafi ocenić przydatność i możliwość wykorzystania nowych osiągnięć (metod i narzędzi) oraz nowych produktów informatycznych [K2st_U6].
Potrafi przeprowadzić krytyczną analizę istniejących rozwiązań technicznych stosowanych w systemach syntezy programów i zaproponować ich usprawnienia [K2st_U8].
Potrafi - wykorzystując m.in. nowe koncepcyjnie metody - rozwiązywać złożone zadania obejmujące projektowanie i realizację systemów syntezy programów, w tym zadania nietypowe oraz zawierające komponent badawczy [K2st_U10].

Kompetencje społeczne
Student rozumie, że w informatyce wiedza i umiejętności bardzo szybko stają się przestarzałe [K2st_K1].
Student rozumie znaczenie wykorzystywania najnowszej wiedzy z zakresu informatyki w rozwiązywaniu problemów badawczych i praktycznych [K2st_K2].

## Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Ocena formatywna:
a) wykłady:
- zadawanie studentom pytań odnoszących się do materiału prezentowanego na poprzednich wykładach.
b) zajęcia laboratoryjne:
- ocena postępów w realizacji projektu.

Ocena podsumowująca:
a) weryfikacja założonych celów dydaktycznych związanych z wykładami:
- Ocena zdobytej wiedzy w formie testu pisemnego, złożonego z kilka pytań otwartych lub zamkniętych. Pytania mają albo charakter teoretyczny (definiować, opisywać, charakteryzować itp.), albo praktyczny (np. symulacja działania algorytmu). Każde pytanie jest warte określoną liczbę punktów, i aby zaliczyć wykład na ocenę pozytywną należy uzyskać przynajmniej 50% możliwych do uzyskania punktów z testu.
b) weryfikacja założonych celów dydaktycznych związanych z zajęciami laboratoryjnymi:
- Ocena, na standardowej skali ocen, realizacji projektu przez studenta, w tym sprawozdania i prezentacji projektu, odbywającej się na ostatnich zajęciach laboratoryjnych, z udziałem pozostałych studentów na widowni. Na ocenę będą miały wpływ takie czynniki jak zasadność przeprowadzonych eksperymentów obliczeniowych, zastosowanego podejścia oraz jakość stworzonego kodu, raportu i prezentacji końcowej.

## Treści programowe

Program modułu obejmuje następujące zagadnienia:
1) Definicja problemu syntezy programów i jego praktyczne zastosowania
2) Specyfikacja zadania syntezy programów
3) Enumeracyjne algorytmy syntezy programów
4) Synteza programów przy użyciu programowania genetycznego
5) Synteza programów przy użyciu sztucznych sieci neuronowych
6) Synteza programów przy użyciu metod rozwiązywania ograniczeń
7) Formalna weryfikacja oprogramowania
8) Weryfikacja modelowa
9) Niezawodność modeli uczenia maszynowego

## Tematyka zajęć

Program wykładu obejmuje następujące zagadnienia:
1) Definicja problemu syntezy programów i jego praktyczne zastosowania
 - wymiary syntezy programów (zdefiniowane przez Sumita Gulwaniego)
 - reprezentacja programów: drzewo składniowe (AST), języki dziedzinowe (DSL)
2) Specyfikacja zadania syntezy programów
 - przykłady wejście-wyjście, specyfikacja logiczna, język naturalny, demonstracja kroków pośrednich
3) Enumeracyjne algorytmy syntezy programów
 - synteza przez unifikację (STUN)
 - algorytm EUSolver
4) Synteza programów przy użyciu programowania genetycznego (GP)
 - reprezentacje programów wykorzystywane w GP: drzewiasta, liniowa, grafowa
 - metody inicjalizacji populacji: full, grow
 - silnie typowane GP
 - ewolucja gramatyk
 - semantyczne GP
5) Synteza programów przy użyciu sztucznych sieci neuronowych
 - sztuczna sieć neuronowa jako metoda priorytetyzacji przeszukiwania przestrzeni programów: DeepCoder
 - grafowe sieci neuronowe w zadaniu syntezy programów
 - podejścia oparte na frameworku wake-sleep: DreamCoder
6) Synteza programów przy użyciu metod rozwiązywania ograniczeń
 - problem spełnialności modulo teorie (SMT)
 - wykorzystanie solwerów SMT do syntezy programów
7) Formalna weryfikacja oprogramowania
 - definicja zadania weryfikacji i przykłady praktycznych zastosowań tej technologii
 - wykorzystanie solwerów SMT do weryfikacji programów
 - logika Hoare'a, niezmienniki pętli
8) Weryfikacja modelowa
 - podstawy logik modalnych
 - logiki epistemiczne
 - logiki temporalne, w tym Computational Tree Logic (CTL)
9) Niezawodność modeli uczenia maszynowego
 - przypadki adwersarialne
 - zagrożenia prywatności w uczeniu maszynowym: radioaktywne dane, zatruwanie danych, wnioskowanie o przynależności, odwracanie modelu, kopiowanie modelu

Na zajęciach projektowych studenci dzielą się na 2-osobowe lub 3-osobowe (w przypadku większych projektów) zespoły i wybierają temat projektu z listy lub proponują własny po konsultacji z prowadzącym. Następnie w trakcie semestru realizują wybrany projekt i okresowo pokazują postępy w pracach. Na ostatnich zajęciach studenci prezentują wykonany projekt i wyniki eksperymentów obliczeniowych przed całą grupą.

## Metody dydaktyczne

Wykłady: prezentacja multimedialna, demonstracja oprogramowania.
Laboratoria: praca zespołowa, konsultacje, prezentacja wyników projektu (oprogramowanie i eksperymenty

obliczeniowe).

## Literatura

Podstawowa
1. Krzysztof Krawiec, "Behavioral Program Synthesis with Genetic Programming", Springer, 2016.
2. Sumit Gulwani, Oleksandr Polozov, Rishabh Singh, "Program Synthesis", Foundations and Trends in Programming Languages, 4(1-2), 2017, 1–119.
3. Wojciech Jamroga, "Logical Methods for Specification and Verification of Multi-Agent Systems", 2015, Monograph Series no 10, ICS PAS Publishing House.

Uzupełniająca
1. Sumit Gulwani. Dimensions in Program Synthesis, Proceedings of the 12th international ACM SIGPLAN symposium on Principles and practice of declarative programming, 2010, 13–24.
2. Rajeev Alur, Rastislav Bodik, Garvit Juniwal, Milo M. K. Martin, Mukund Raghothaman, Sanjit Seshia, Rajdeep Singh, Armando Solar-Lezama, Emina Torlak, Abhishek Udupa. Syntax-Guided Synthesis, Formal Methods in Computer-Aided Design (FMCAD), IEEE, 2013, 1–8.
3. Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, Daniel Tarlow. DeepCoder: Learning to Write Programs, Proceedings of the International Conference on Learning Representations, 2017.
4. Armando Solar-Lezama, Liviu Tancau, Rastislav Bodík, Sanjit A. Seshia, Vijay A. Saraswat. Combinatorial Sketching for Finite Programs, Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems, 2006, 404–415.
5. Susmit Jha, Sumit Gulwani, Sanjit A. Seshia, Ashish Tiwari. Oracle-Guided Component-Based Program Synthesis, In 2010 ACM/IEEE 32nd International Conference on Software Engineering, volume 1, 2010, 215–224.
6. Swarat Chaudhuri, Kevin Ellis, Oleksandr Polozov, Rishabh Singh, Armando Solar-Lezama, and Yisong Yue. Neurosymbolic Programming. Foundations and Trends in Programming Languages, 7, 3 (Dec 2021), 158–243. https://doi.org/10.1561/2500000049.

## Bilans nakładu pracy przeciętnego studenta

| | Godzin | ECTS |
|---|---|---|
| Łączny nakład pracy | 100 | 4,00 |
| Zajęcia wymagające bezpośredniego kontaktu z nauczycielem | 60 | 2,00 |
| Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu) | 40 | 2,00 |