Jit Team™

Współczesne Zastosowania Informatyki

Bio-Inspired Computing

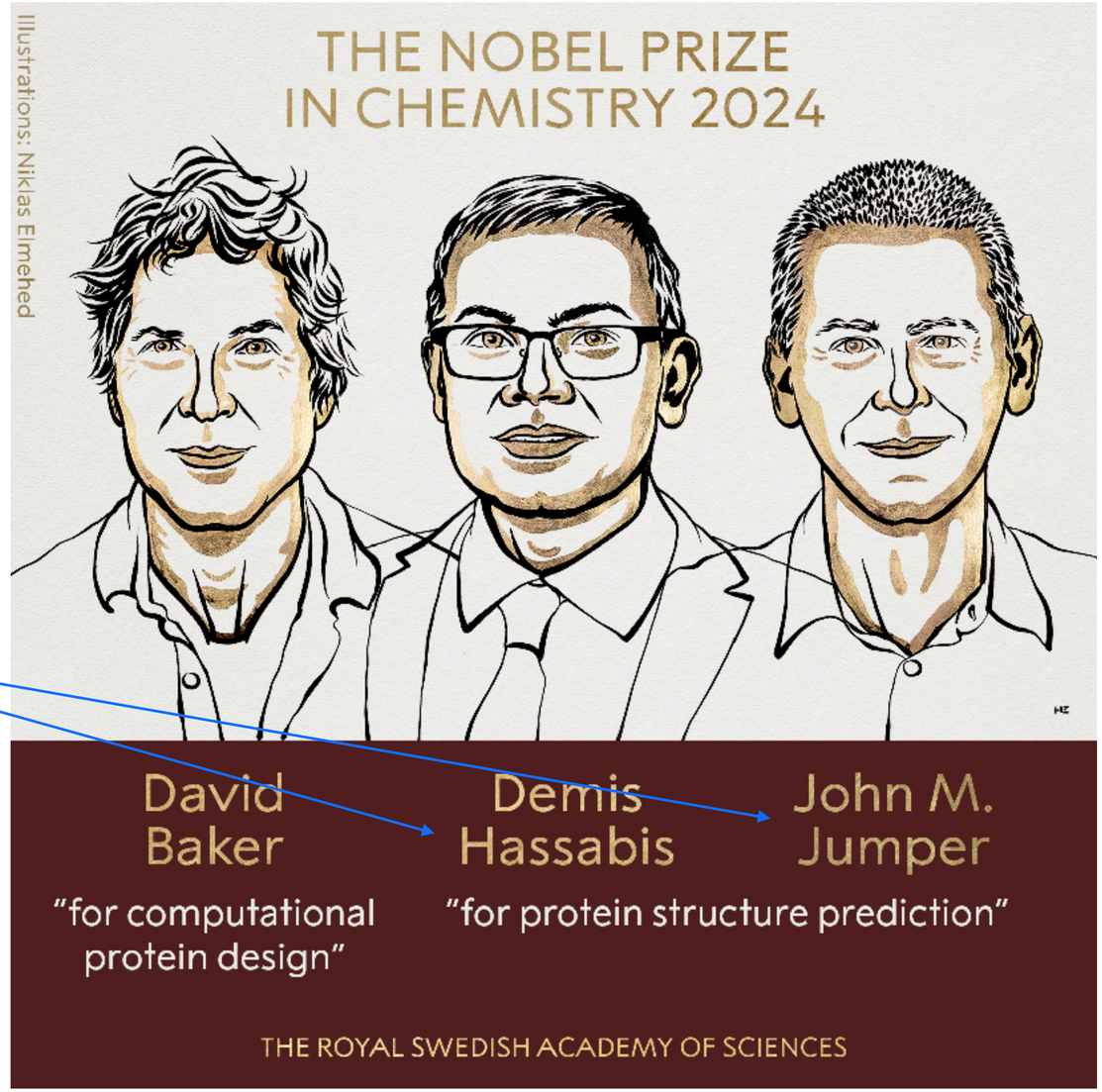
Wykład #2

dr Witold Bołt, 16.10.2024 r.

Nobel z Chemi

Google DeepMind

AlphaFold, 2020



Ogłoszenia

- Nie ma jeszcze grupy na Teams :(ale będzie

- Za tydzień 23 października 2024 r. - zajęcia **odwołane** (zarówno **wykład** jak i **laboratorium**)
  - Osoby, które w tym dniu zgodnie z planem powinny mieć laboratoria **wysyłają** swoje zadania **e-mail**. Zajęcia “*przepadają*”. Ocenię prace offline i napiszę każdemu feedback mailem.

Wstępna rozpiska wykładów

#	Data	Prowadzący	Firma	Temat
-	02.10.2024	**ODWOŁANE**		
1	09.10.2024	Witold Bołt	Jit Team	Wprowadzenie
2	16.10.2024	Witold Bołt	Jit Team	Bio-Inspired Computing
-	23.10.2024	**ODWOŁANE**		
3	30.10.2024	Paweł Noga	freelance	AI
4	6.11.2024	Krzysztof Kąkol	Xebia	Jak działa ChatGPT
5	13.11.2024	Marek Karpiński	Gemini	
6	20.11.2024	Paweł Góra	Center for Computational Medicine	
7	27.11.2024	wolny termin		
8	4.12.2024	Michał Bojko	Dynatrace	
9	11.12.2024	Filip Drzewiecki	Jit Team	
10	18.12.2024	Marek Zmuda	Intel	
11	8.01.2025	Stanisław Matczak	SmartRecrutiers	
12	15.01.2025	wolny termin		
13	22.01.2025	Damian Kobiałka	ColoredTree	
14	29.01.2025	Krzysztof Radecki	DAC.digitial	Blockchain

Bio-inspired Computing

… kiedy natura pomaga informatyce

Algorytmy Genetyczne

Automaty Komórkowe

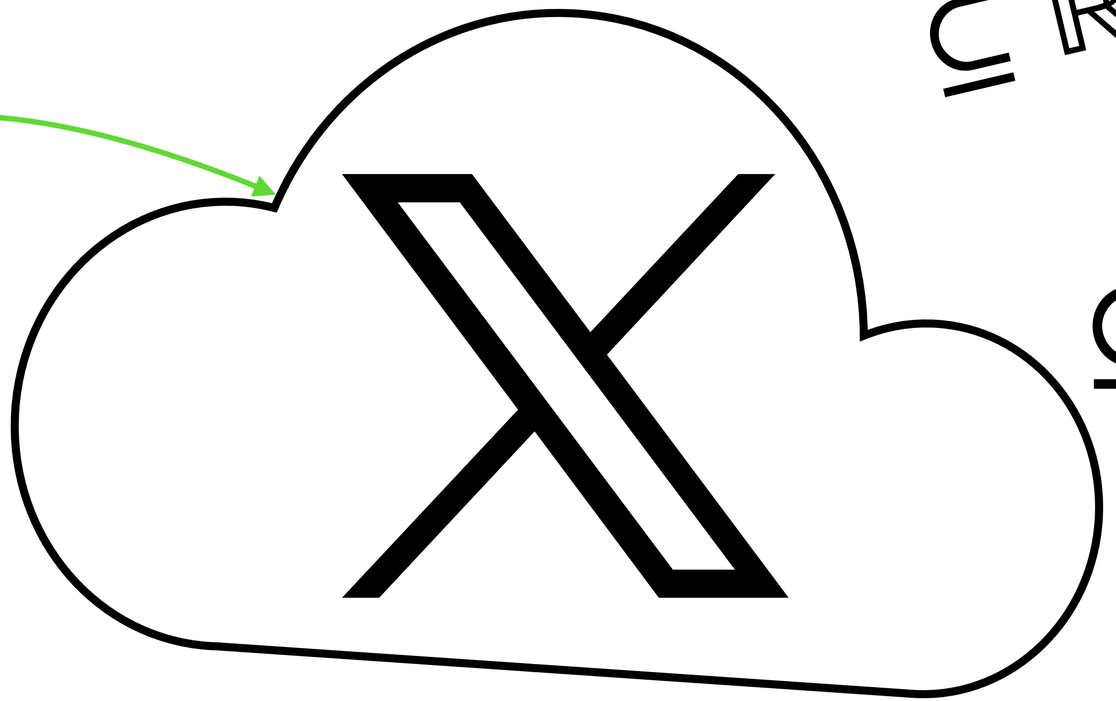
Algorytmy Genetyczne / Ewolucyjne

Przestrzenie wektorowe

$\subseteq \mathbb{R}^n$

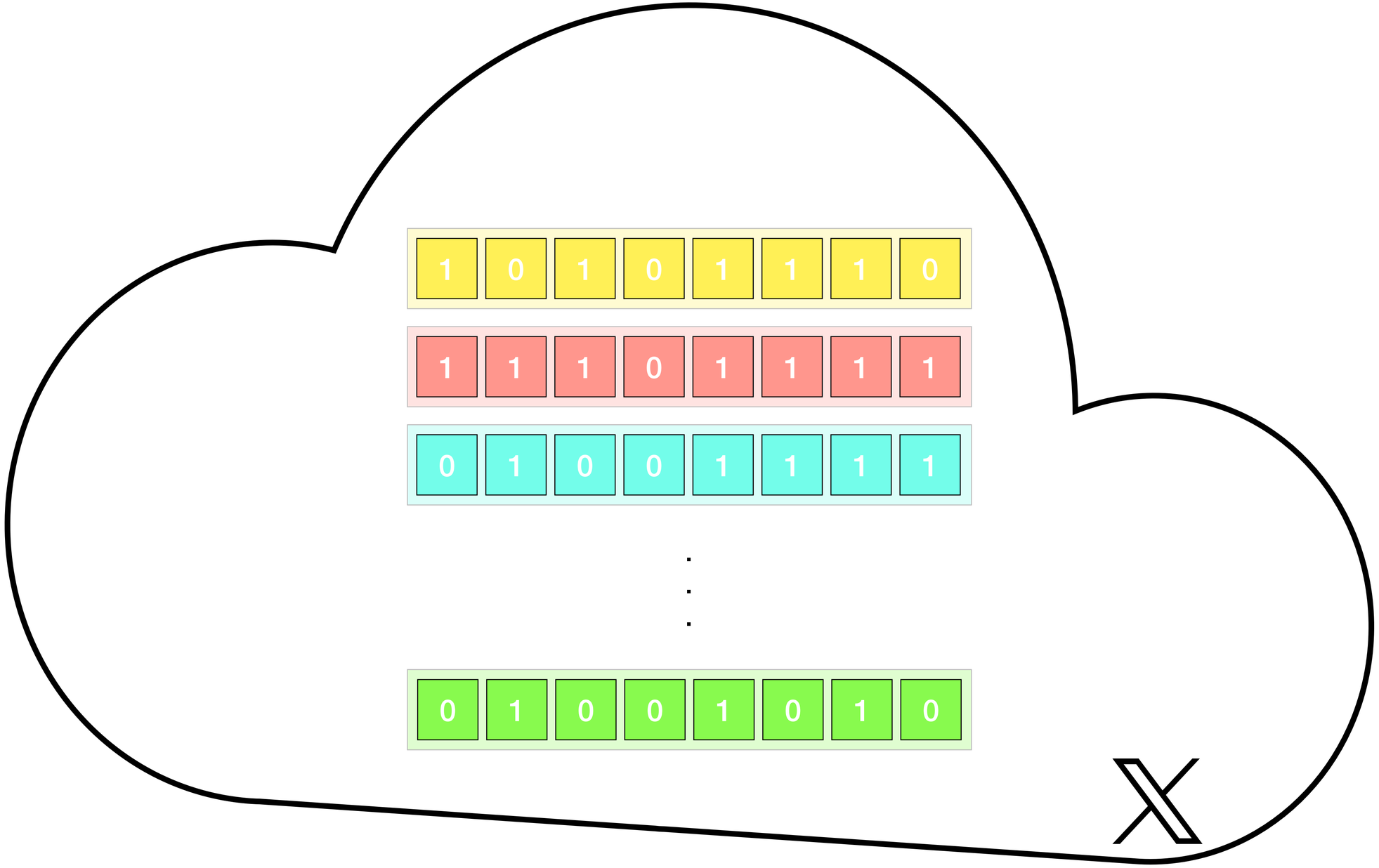
$\subseteq [0,1]^n$

$\subseteq \{0,1\}^n$



search space

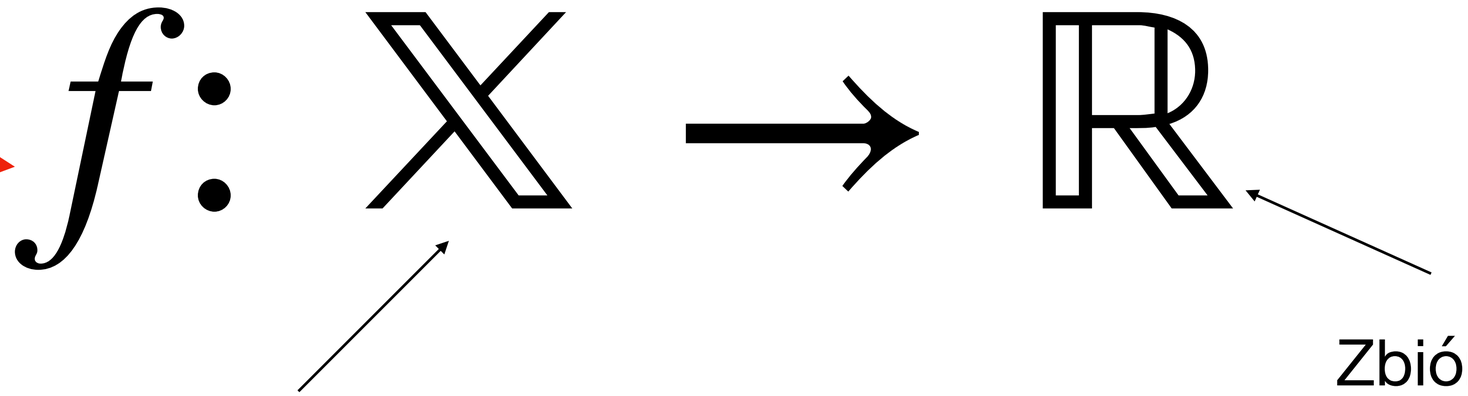
Przestrzeń w której poszukujemy ***rozwiązania***.



Przykład. Wektory binarne długości $n = 8$. Czyli $\mathbb{X} \subseteq \{0,1\}^8$.

Fitness function

Funkcja dopasowania

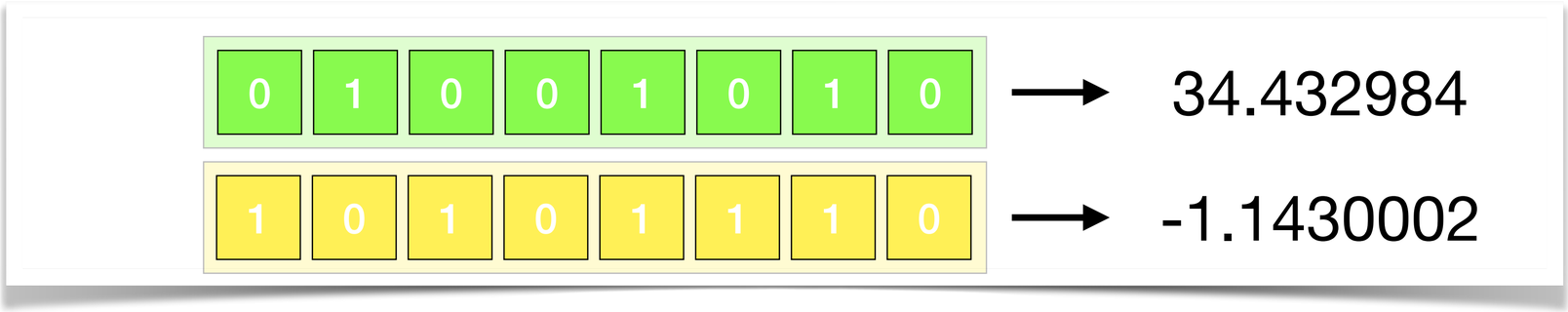


Dziedzina

Wektory…

Zbiór wartości

Zwykłe **liczby** :)



Kto ustala $\mathbb{X}$ oraz f? Skąd je wziąć?

Sami wymyślamy! Jak? O tym później…

W szczególności może **nie** być ogólnego wzoru funkcja f a jedynie **procedura** liczenia wartości.

Problem **optymalizacji**. Szukamy takiego $x \in \mathbb{X}$ dla którego $f(x)$ jest **największe**.

(… moglibyśmy policzyć pochodną, albo gradient…
*ale **nie wiadomo** czy się da!)*

Innymi słowy szukamy **maksimum** funkcji f,

… ale niestety nie zakładamy **nic** o tej funkcji f

Black box optimization.

Zadanie. Jak optymalnie ustawić antenę?

wartość funkcji dopasowania
siła odbieranego sygnału

konkretne ustawienie pokręteł
wektor z przestrzeni $\mathbb{X}$

panel sterowania anteny

Pomysł! Zainspirujmy się przyrodą…

- Traktujemy elementy zbioru $\mathbb{X}$ (*wektory*) jak **żywe organizmy**, a wartości funkcji f jako “współczynnik” **przystosowania** do otoczenia.
  - Żywe organizmy **rozmnażają się**.
  - Lepiej dostosowane osobniki mają większe szanse na **przeżycie** i wydanie **potomstwa**. Lepiej dostosowane osobniki są bardziej **atrakcyjne**.
  - W procesie rozmnażania następuje **dziedziczenie** cech - tzn. “potomek” ma część cech jednego “rodzica”, część cech drugiego “rodzica”.
  - Poza semi-deterministycznym procesem dziedziczenia cech, w życiu jest też szansa na **przypadek** - mutacje / zaburzenia, które wprowadzają **nowe** cechy nieistniejące u “rodziców”.
  - Żywe organizmy również się odżywiają, poruszają itd. Ale te aspekty życia **pomijamy**.

Algorytmy Genetyczny / Ewolucyjny

- Będziemy pracować ze **skończonymi** podzbiorami $\mathscr{P} \subset \mathbb{X}$, które będziemy nazywać populacjami albo generacjami. Pierwsza - **inicjalna populacja** powstaje przez wylosowanie określonej liczby $k > 0$ elementów z $\mathbb{X}$. (Póki co nie ma tu nic mądrego.) Inicjalną populację oznaczymy przez $\mathscr{P}_0$.

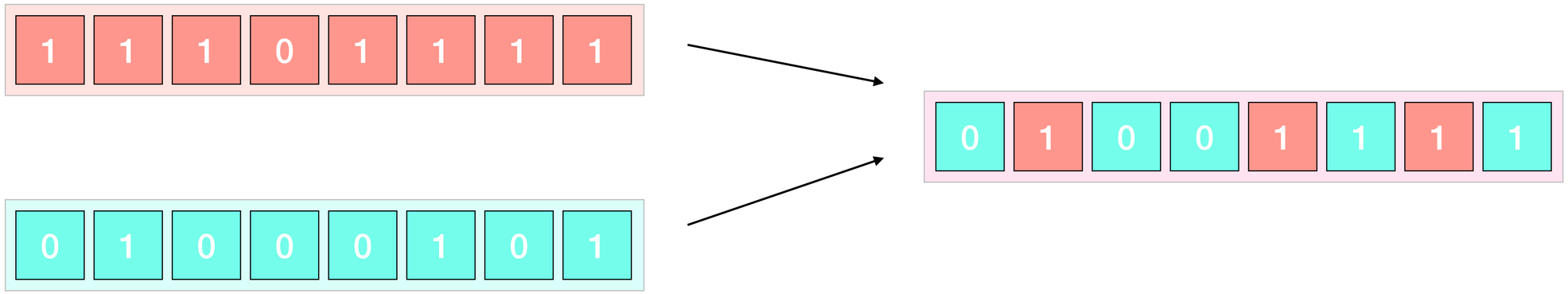
- Główny krok algorytmu to przepis na budowę populacji $\mathscr{P}_i$ **na podstawie** populacji $\mathscr{P}_{i-1}$.

- Odpowiada to (*bardzo luźno*) procesowi następowania po sobie kolejnych pokoleń prostych organizmów żywych.

Budowanie populacji

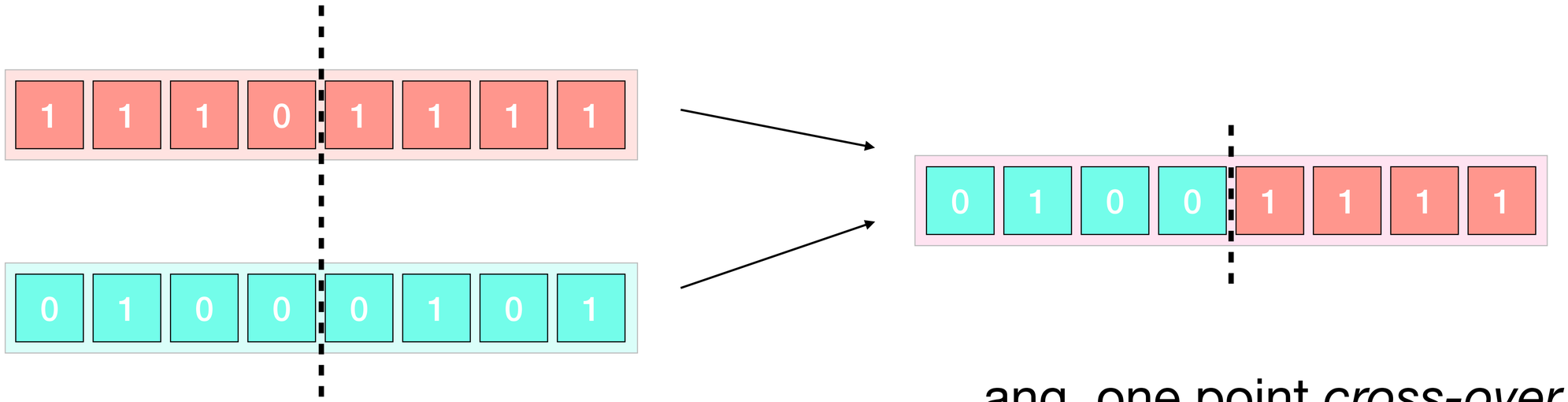
- Załóżmy, że $\mathscr{P}_{i-1}$ jest nam znana. Będziemy budować $\mathscr{P}_i$.
- Liczymy wartości funkcji dopasowania dla osobników $\mathscr{P}_{i-1}$ i zapisujemy.
- Powtarzamy k razy:
  - Losujemy dwa elementy z $\mathscr{P}_{i-1}$, niezależnie ze zwracaniem, w taki sposób, że prawdopodobieństwo wyboru danego elementu jest *proporcjonalne* do wartości jego funkcji dopasowania (*czym wyższa wartość tym wyższa szansa wyboru*). [operator **selekcji**]
  - Łączymy dwa wybrane elementy ucinając wektory w losowych miejscach i sklejając je. [operator **krzyżowania**]
  - Aplikujemy drobne zmiany na losowo wybranych pozycjach wyniku. [operator **mutacji**]
  - Tak utworzony element dodajemy do $\mathscr{P}_i$.
- **Opcjonalnie**. Zapewniamy, że ustalona liczba “najlepszych” względem wartości dopasowania elementów z P_{i-1} przechodzi do $\mathscr{P}_i$ **bez zmian**. [operacja zachowania elit - **elite survival**]

Krzyżowanie jednorodne



ang. *uniform cross-over*

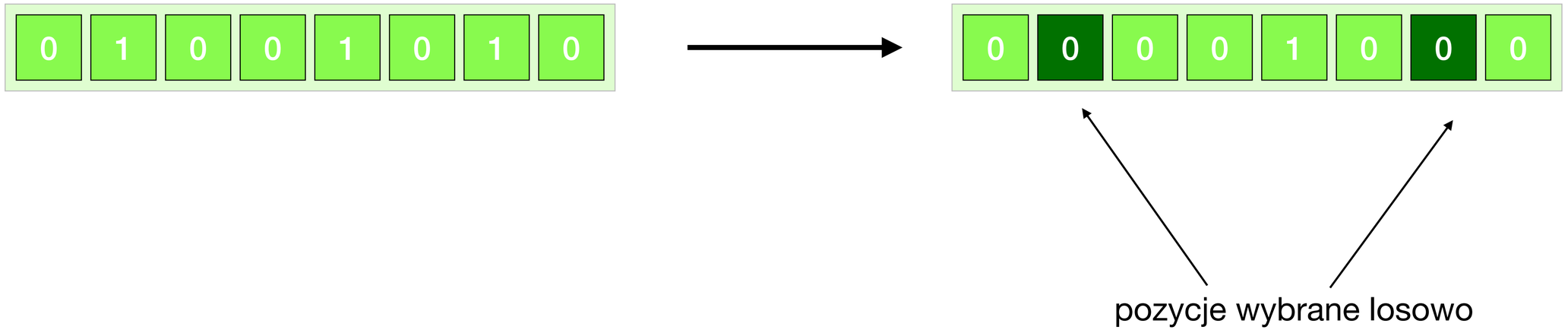
Krzyżowanie jednopunktowe



ang. one point *cross-over*

Możliwy też wariant dwu punktowy…

Mutacja



Okazuje się, że to wystarczy!

Przykład: rozwiązywanie równań

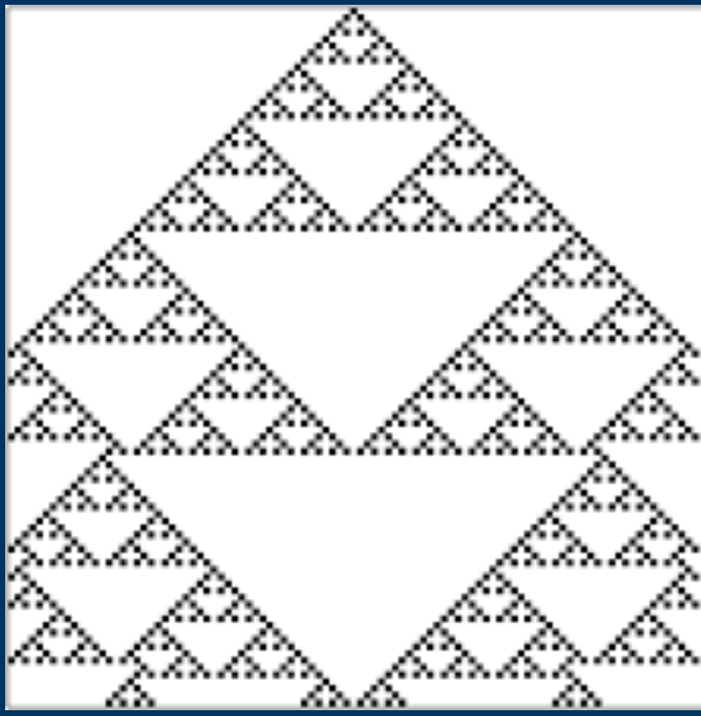
https://github.com/houp/ca-class/blob/main/simple_ga.ipynb

Co z tego wynika? Co tu widzimy?

- Bardzo prosta **sztuczna inteligencja**!
  - Program, który potrafi rozwiązywać szeroką klasę problemów, bez konieczności projektowania go “pod konkretny problem”.

- Możliwość stosowania w różnych dziedzinach. Również bardzo praktycznych.
  - Można stosować do znajdowania sieci neuronowych (**neuroevolution**)!
- Nie zawsze jest najlepszym rozwiązaniem. Dlaczego? Bo czasem wiemy coś więcej o funkcji f i możemy tą wiedzę wykorzystać (np. umiemy policzyć gradient i wtedy możemy stosować podejścia oparte o *gradient descent*).

Automaty Komórkowe



Demo: Game of Life

https://playgameoflife.com/

https://golly.sourceforge.io/

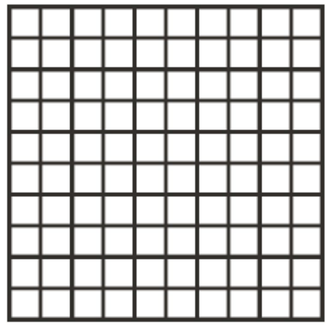
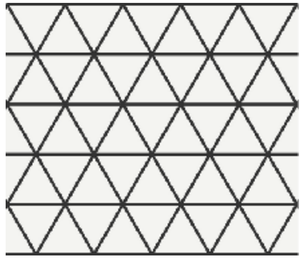
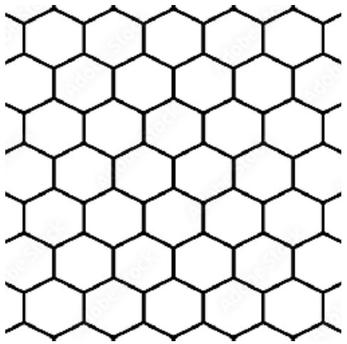
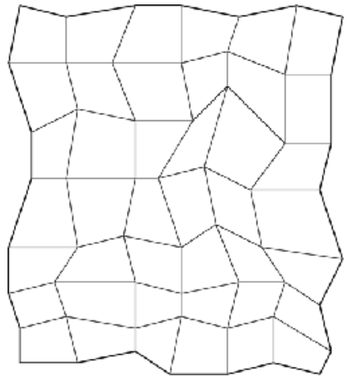
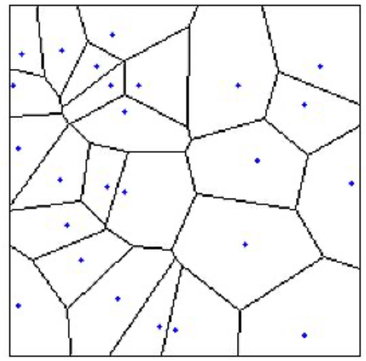
Automaty komórkowe (cellular automata)

- Przestrzeń podzielona na **komórki** (niepodzielne, **dyskretne** elementy - kawałki przestrzeni). *Świat jest “w kratkę” - jak w Minecraft.*
- W danej chwili czasu komórka jest w określonym **stanie**. Zbiór stanów jest **skończony** (np. dwu elementowy).
- Stany podlegają ewolucji (zmianom) w czasie. Czasie jest **dyskretny**.
- Zmiana stanu odbywa się zgodnie z **deterministyczną** regułą lokalną.
- Stan komórki w chwili $t + 1$ zależy od jej stanu w chwili t oraz od stanów jej **sąsiadów** w chwili t.

Co zrobią aby zaprojektować automat komórkowy?

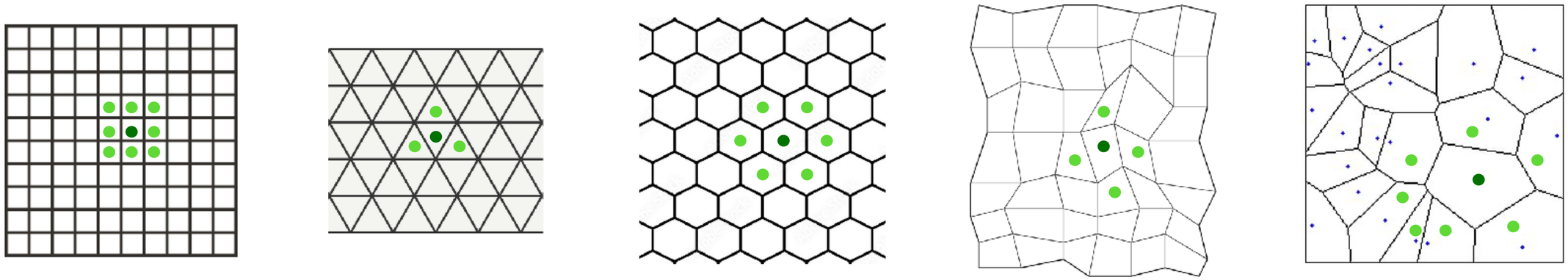
- Ustalić zbiór stanów. Ile elementów i jakie?

- Wybrać przestrzeń i podzielić ją na komórki - np. przestrzeń dwu-wymiarowa podzielona na “kratkę”.

Co zrobią aby zaprojektować automat komórkowy?

- **Sąsiedztwo** - musimy wskazać co to znaczy, że komórka ma sąsiadków.

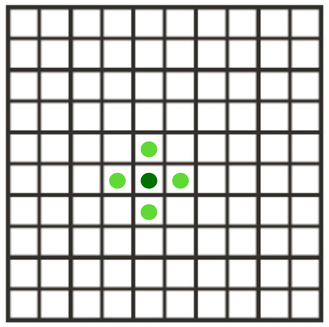


komórka “centralna”

komórka “sąsiednie” względem “centralnej”

Co zrobią aby zaprojektować automat komórkowy?

- **Reguła lokalna** — funkcja która, dla danej **konfiguracji sąsiedztwa** (czyli uporządkowanych stanów w sąsiedztwie) wyliczy nowy stan.



$$f\begin{pmatrix} & x_u & \\ x_l & x_c & x_r \\ & x_d & \end{pmatrix} \longrightarrow f(x_u, x_l, x_c, x_r, x_d)$$

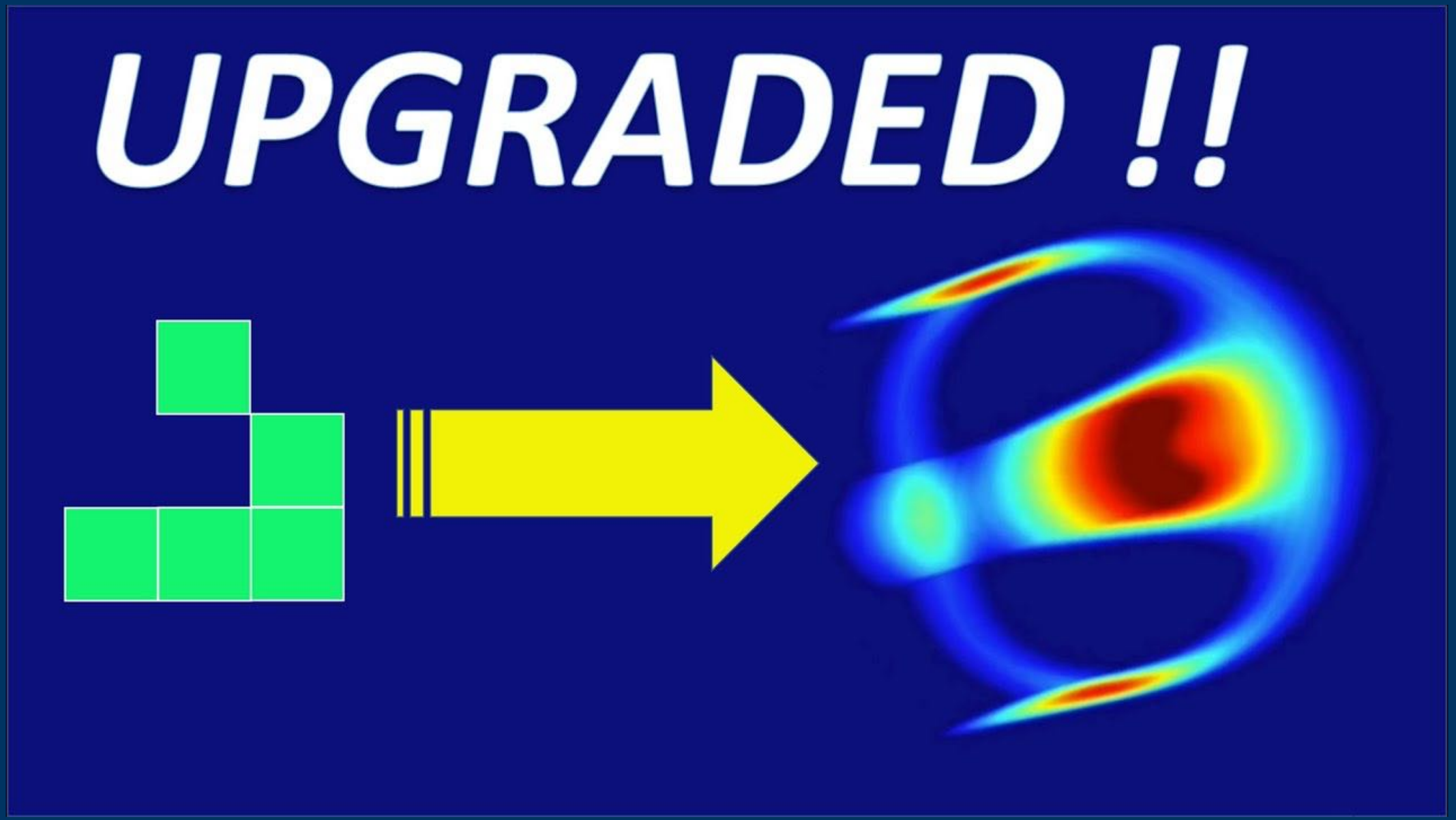
Przykłady modeli CA

Załącznik: wykład #11 z przedmiotu CA

https://github.com/houp/ca-class/blob/main/slides/lecture11.pdf

Co wynika z Automatów Komórkowych

- Proste modele złożonych procesów naturalnych
- Universal computation
- Emergent behavior - “całość jest większa niż suma części”
- Prosta definicja - skomplikowane własności



https://www.youtube.com/watch?v=7-97RhAZhXI

Dziękuję!

Witold.Bolt@ug.edu.pl

Współczesne
Zastosowania
Informatyki

Jit Team™